WL-TR-94-4002

INTELLIGENT AUTOMATED PROCESS
PLANNING AND CODE GENERATION FOR
COMPUTER-CONTROLLED INSPECTION

# AD-A275 346

STEVEN M. RUEGSEGGER

CASE WESTERN RESERVE UNIVERSITY
ELECTRICAL ENGINEERING AND APPLIED
   PHYSICS
CLEVELAND OH   44106-7721

JANUARY 1994

FINAL REPORT FOR 08/01/91-01/01/93

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

MATERIALS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT PATTERSON AFB OH 45433-7734
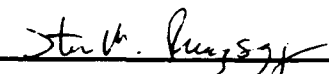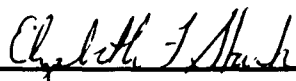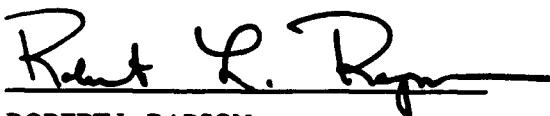
94-03099

94 1 31 200

# NOTICE

WHEN GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA ARE USED FOR ANY PURPOSE OTHER THAN IN CONNECTION WITH A DEFINITELY GOVERNMENT-RELATED PROCUREMENT, THE UNITED STATES GOVERNMENT INCURS NO RESPONSIBILITY OR ANY OBLIGATION WHATSOEVER. THE FACT THAT THE GOVERNMENT MAY HAVE FORMULATED OR IN ANY WAY SUPPLIED THE SAID DRAWINGS, SPECIFICATIONS, OR OTHER DATA, IS NOT TO BE REGARDED BY IMPLICATION, OR OTHERWISE IN ANY MANNER CONSTRUED, AS LICENSING THE HOLDER, OR ANY OTHER PERSON OR CORPORATION; OR AS CONVEYING ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY IN ANY WAY BE RELATED THERETO.

THIS REPORT IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

STEVEN M. RUEGSEGGER
Research Engineer, Manufacturing Research
Integration and Operations Division
Materials Directorate

ELIZABETH F. STARK, Capt, USAF
Acting Branch Chief, Manufacturing Research
Integration and Operations Division
Materials Directorate

ROBERT L. RAPSON
Chief, Integration and Operations Division
Materials Directorate

IF YOUR ADDRESS HAS CHANGED, IF YOU WISH TO BE REMOVED FROM OUR MAILING LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION PLEASE NOTIFY WL/MLIM, WRIGHT-PATTERSON AFB, OH 45433-7746 TO HELP MAINTAIN A CURRENT MAILING LIST.

COPIES OF THIS REPORT SHOULD NOT BE RETURNED UNLESS RETURN IS REQUIRED BY SECURITY CONSIDERATIONS, CONTRACTUAL OBLIGATIONS, OR NOTICE ON A SPECIFIC DOCUMENT.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | January 1994 | Final August 1991 – January 1993 |

| 4. TITLE AND SUBTITLE | 5. FUNDING NUMBERS |
|---|---|
| Intelligent Automated Process Planning and Code Generation for Computer-Controlled Inspection | C: F33615-87-C-5250 |
| **6. AUTHOR(S)** | PE: 62102F |
| | PR: 2306 |
| | TA: P9 |
| Ruegsegger, Steven M | WU: 03 |

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|
| Case Western Reserve University<br>Electrical Enginering and Applied Physics<br>Cleveland OH 44106-7721 | |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
|---|---|
| Materials Directorate<br>Wright Laboratory<br>Air Force Material Command<br>Wright Patterson AFB OH 45433-7746 | WL-TR-94-4002 |

**11. SUPPLEMENTARY NOTES**

Master's Thesis of author

| 12a. DISTRIBUTION/AVAILABILITY STATEMENT | 12b. DISTRIBUTION CODE |
|---|---|
| Approved for public release; distribution is unlimited | |

**13. ABSTRACT** *(Maximum 200 words)*

Requirements for greater precision and reduced rejection rates demand improved inspection methods that can be provided by implementing increased automation into the process. This thesis discusses the implementation of an automated intelligent inspection planner and its integration into a feature-based concurrent engineering system. The approach utilizes features as the common language of the individual modules that promote ideas of geometry, functionality, and design intent throughout the system by feature translation among the modules. An artificial neural network optimizes the sequence of inspection points based on inspection rule criteria. A collision avoidance algorithm ensures the safety of automated inspection in a computationally efficient manner. The goal of the inspection planner is to output instruction code that will be executed on a computer-controlled coordinate measurement machine (CMM) to properly, efficiently, and safely measure and evaluate the tolerances of the manufactured product.

| 14. SUBJECT TERMS | | | 15. NUMBER OF PAGES |
|---|---|---|---|
| Automated Inspection Neural Networks Computer Aided Process Planning | | | 149 |
| | | | **16. PRICE CODE** |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

to my wife, father, and angel mother

# Preface

All figures and text results in this thesis are actual results produced by the methods described in this thesis.

If you would like more information about this thesis, the Rapid Design System, or would like information on how to receive this thesis in a postscript text file, please send an Internet e-mail request to smr2@po.cwru.edu.

| Accesion For | | |
|---|---|---|
| NTIS CRA&I | ☒ | |
| DTIC TAB | ☐ | |
| Unannounced | ☐ | |
| Justification | | |
| By | | |
| Distribution / | | |
| Availability Codes | | |
| Dist | Avail and / or Special | |
| A-1 | | |

DTIC QUALITY INSPECTED 8

iv

# Acknowledgments

I would like to give a very special thanks to Dr. Frank Merat for whom I have enough respect to call "Dr. Merat," yet to whom I also feel close enough to call "Frank." He has been my advisor for about five years. He has given good, honest, and fair advice for which I will always be thankful.

My fellow students have provided much assistance. Bob "rbd" Delvalle has shown incredible patience to put up with my current crises of the day. Kavous "Kyous" Roumina gave much overall direction of the IPEM. Alok "take-a-look" Mathur was always willing to stop what he was doing to watch me figure it out for myself. Natarajan "Yeah, I don't know" Balasundara and Leda "See ya Leda" Villalobos also provided assistance whenever asked of them.

A special note of thanks goes to Dr. Yoh-Han Pao. His research excellence provided the link to the USAF which created this project and provided the computer facilities.

Thanks is also due to the men at Wright-Patterson Air Force Base, MLIM, of whom I am now a colleague. Dr. Walt Griffith and Dr. Steve LeClair have headed the Rapid Design System research, and Dr. Gerry Radack guided every module of the RDS to its present state, providing consistency to the separated group of researchers.

A final note of appreciation goes to my family for their love, patience, and prayers. My wife Julie has demonstrated true Agape spirit throughout the long hours of research and writing. Her father, Fred Scheffler, has worked diligently with the technical editing of this thesis, making it readable.

# Table of Contents

# List of Illustrations

# List of Tables

# List of Abbreviations

2-D ................ Two Dimensional
3-D ................ Three Dimensional
AI ................... Artificial Intelligence
ANN .............. Artificial Neural Network
ANSI ............. American National Standard Institute
CAD .............. Computer Aided Design
CAM ............. Computer Aided Manufacturing
CAI ................ Computer Aided Inspection
CAPP ............ Computer Aided Process Planning
CIM ............... Computer Integrated Manufacturing
CLOS ............ Common LISP Operating System
CMES ............ Coordinate Measurement Software
CMM ............. Coordinate Measurement Machine
DMIS ............ Dimensional Measurement Interface Specification
dof ................. degree(s) of freedom
DRF ............... Datum Reference Frame
EAM .............. Episodal Associative Memory
FBDE ............ Feature Based Design Environment
FAB-PLAN ... Fabrication Planning
GD&T ........... Geometric Designing and Tolerancing
IPEM ............. Inspection Planning and Evaluation Module
IPF ................ Inspection Plan Fragment
LMC .............. Least Material Condition
MMC ............. Maximum Material Condition
MR ................ Measurement Request
OOP .............. Object Oriented Programming
QA ................. Quality Assurance
RDS ............... Rapid Design System
RFS ............... Regardless of Feature Size
TSP ............... Traveling Salesman Problem

# Chapter 1

# Introduction

*People who like this sort of thing*
*will find this the sort of thing they like*
                                    - Abraham Lincoln

---

The task of the Quality Assurance (QA) engineer is to determine if the geometries of the product are within the specified tolerances created by the design engineer. The results should determine whether or not the product will perform its desired functions correctly. Today's technologies have provided a diverse range of automated inspection systems for QA. The primary measurement technique used in automated industrial inspection of machined parts is the coordinate measurement machine [Galm, 1991] [Menq, et al., 1991a] [ElMaraghy and Gu, 1987]. Advanced graphical programming tools have also simplified some aspects of the automated inspection process. Computer-aided design, process planners, path planners, and simulators provide assistance to the inspector.

This thesis discusses the implementation of an automated inspection planner operating within a feature-based concurrent engineering system called the Rapid Design System (RDS). The RDS contains software modules to automate the design, manufacture, and inspection aspects of product fabrication, as well as an artificial intelligence memory to provide advanced storage and retrieval of designs. The automated inspection planner interfaces to the CAD system to receive the design,

1

tolerance, and function intent information. The task of the automated inspection planner is to produce the inspection plan which properly evaluates the tolerances on the manufactured product. Important aspects of an automated inspection plan include a safe, collision-free, and efficient path trajectory throughout the probe-space. The output is generated as actual instruction code that controls the execution of the inspection plan on a computer-controlled coordinate measurement machine. The output is complete and will require no human editing (or at least, no more than fine-tuning or "tweaking") before the code can be executed.

The individual accomplishments of this thesis introduced several important elements into the automated inspection planner of the RDS:

1. An artificial neural network performs sequence optimization on the inspection points using an inspection rule based criterion.

2. A computationally efficient collision-avoidance algorithm creates a safe path for the probe head to travel around the workpiece.

3. An automated code generator produces the CMM instructions that will safely, efficiently, and correctly evaluate the tolerance on the workpiece.

### Thesis Overview

Chapter 2 provides important background information and a literature review to provide better understanding of the work described in this thesis as well as relate it to research in similar areas. Since the work of this thesis is a sub-part of a concurrent engineering system, some aspects of the larger project are discussed to place proper perspective of how this thesis fits into the intentions of the engineering system. The engineering tools and techniques of object oriented programming, the feature and hierarchical paradigms, and concurrent engineering are also described. Chapter 3 discusses the theme of the research: quality assurance. The standards, formats, and

techniques of quality assurance used in this research are described. They include the ANSI Y14.5M-1982 tolerancing standard, coordinate measurement machines, and automated inspection.

Chapter 4 discusses the implementation of the intelligent automated inspection planner. The work of several researchers (see Acknowledgments) and their contribution to this thesis is developed in sections 4.1 and 4.2.1. Sections 4.2.2 through 4.3 describe the individual research and implementation of work performed to achieve this thesis. An example of a product's representation at each stage throughout the inspection planner is illustrated in Chapter 5.

This thesis closes with conclusions and future work in chapters 6 and 7, respectively. Appendix A shows the inspection language macros that create the output of the inspection planner. Appendix B gives a quick reference to the inspection language of the coordinate measurement machine owned by the end-user of the Rapid Design System. It can be used to clarify the examples and to give the reader an understanding of one reason for the necessity of an automated inspection planner. The inspection rule-based artificial neural network program code is presented in Appendix C.

# Chapter 2

# Background

*Those who cannot remember the past*
*are condemned to repeat it.*
*- George Santayana*

---

This chapter describes the factors that influenced this research. The larger research project of the concurrent engineering system defined the project platform and software implementation language. A problem and approach were constructed by studying the efforts of the QA engineers at the 4950th Test Wing, Wright-Patterson Air Force Base as they performed their job of inspecting manufactured parts using their newly acquired CMM. A literature review helped guide this research by introducing new ideas that could be developed further. Finally, background topics critical to this research effort are explained and defined.

## 2.1. Project Background

This work is encompassed by a project funded by the U.S. Air Force to develop an intelligent expert system which reduces the turnaround time of a product from design to manufacture and inspection. This system, called the Rapid Design System (RDS), is being developed with the cooperation of an Air Force design and manufacturing Test Wing [LeClair, 1991]. This organization specializes in the custom design

and manufacture of aircraft replacement parts which are no longer available from the original manufacturer. Therefore, the lot sizes are very small, typically ranging from one to twenty. For these small batch sizes, the highest time factor is the lead time in product plan development in both the manufacturing and inspection arenas. The objective of the RDS is to drastically reduce this lead time by providing information links throughout the life cycle of the product. Both the manufacturer and inspector receive the entire feature-based part-model that the designer created, rather than an engineering drawing or a simple surface representation. This allows the process planner to use all the function and tolerance intent inherent in the design and tolerance features.

The RDS is built upon the Concept Modeler™, a parametric design system from Wisdom Systems, Inc. The Concept Modeler™ is built upon Common LISP Operating System (CLOS), an object-oriented programming language with inheritance. The RDS uses these platforms to provide features used throughout the concurrent engineering system.

The four RDS modules are as follows: the Episodal Associative Memory, the Feature-Based Design Environment, the Fabrication Planning module, and the Inspection Planning and Evaluation module. The overall purpose of the RDS is to provide a common language that allows these different modules to be interconnected so that process planning and other artificial intelligence heuristics can use the translated data. Its goal is to relieve the manufacturer or inspector from the guess-work often involved in interpreting the tolerancing and functional design intent of a product once it is "thrown over the engineering wall."

**Rapid Design System**
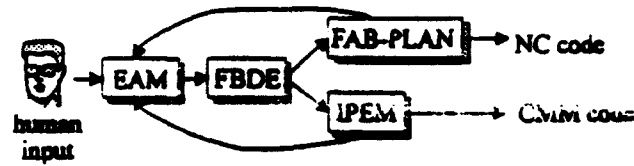


**Figure 1.** Rapid Design System overview.

The *Episodal Associative Memory* (EAM) is a computer-based associative memory, whose purpose is to augment the human designer's memory by providing an institutional or collective memory for all phases of the product's life cycle [Pao, et al., 1991]. Information regarding the successes and failures of a product from each of its production stages is stored in the memory. The life cycle memory includes information from the designer, manufacturer, and inspector. This experience data of similarly designed parts is then "remembered" by the memory when a new part is introduced into the system. The goal is to avoid the costly mistakes that occurred previously and to reiterate positive factors of all aspects of the similar product's life cycle. The designer can learn from the trials and solutions of the fabrication or inspection engineers. These trials and solutions might not otherwise be considered due to job turnover, promotions, forgetfulness, disorganization, misplaced or destroyed papers, etc.

The *Feature-Based Design Environment* (FBDE) is the front-end CAD system where the designer creates the product. In the feature-based design paradigm (section 2.5), the product, or part model, is described in terms of features which represent higher-level concepts than the geometric primitives used in traditional CAD systems [Radack, et al., 1991]. Most commercially available computer-aided drafting systems

are wire-frame representations which depict objects by geometric primitives: points, lines, curves, circles, etc. Tolerancing, whether it is the ANSI geometric standard or, more commonly, the traditional rectangular (±) standard, is accomplished by placing text and an arrow in the drawing, not linking it in any way to the actual geometry of the part model.

The FBDE uses Constructive Solid Geometry (CSG) to represent the 3-D part model. One class of features, called "form-features," contains both negative volume features such as holes and pockets, as well as positive volume features such as bosses and ribs. When attached to a feature or sub-feature, negative volume features will remove material, while positive volume features will add material. Another class of features serves to modify the geometry of the form feature, e.g., chamfers and fillets.
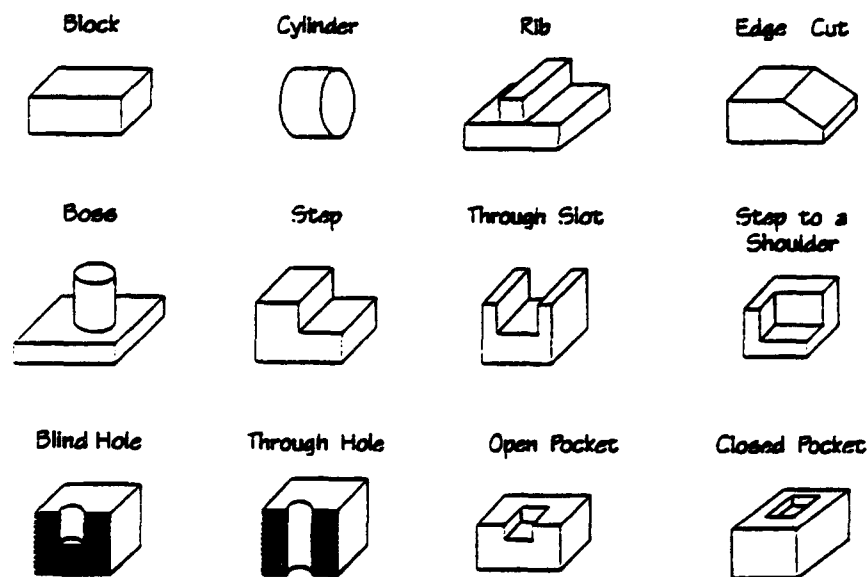


Figure 2. Design features implemented within the RDS.

The FBDE also represents dimensioning and tolerancing information within the part model using the feature paradigm. The representation conforms to the ANSI

Y14.5M standard called Geometric Dimensioning and Tolerancing (GD&T), e.g.,

straightness, position, and flatness. These tolerance features are attached to geomet-

ric features or surfaces of the part model representing the proper GD&T callout pro-

cedure, as well as needed information for the process planner.

| Simple Datum | Basic Dimension | Datum Reference Frame | Flatness |
| Position | Straightness | Parallelism | Perpendicularity |
| Concentricity | Cylindricity | Circularity | Angularity |
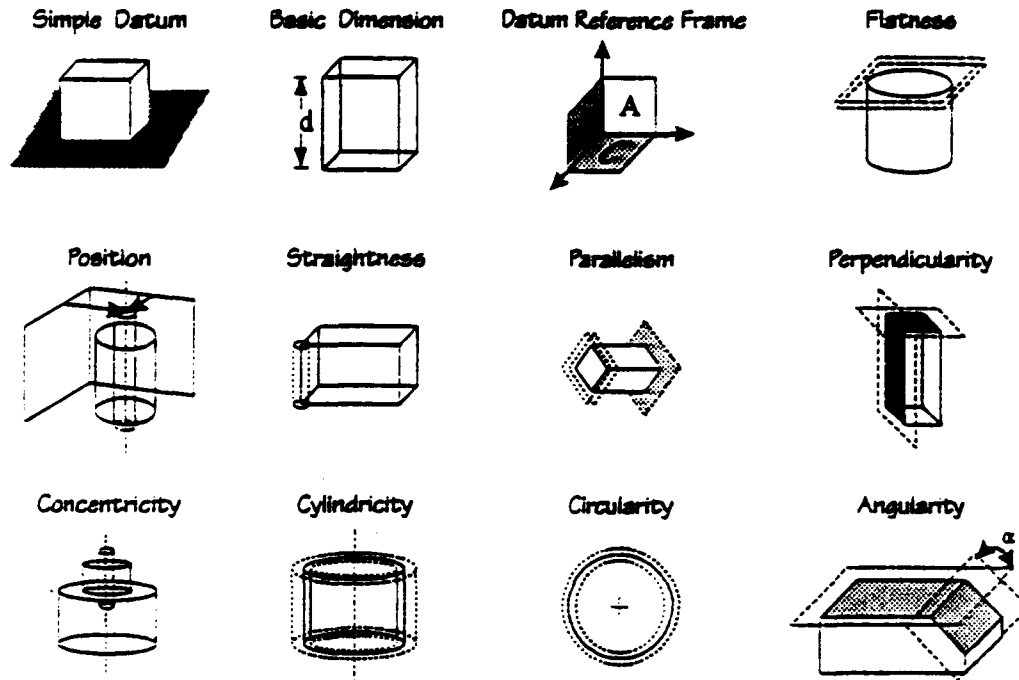
Figure 3. Tolerance features implemented within the RDS.

*Fabrication Planning* (FAB-PLAN) outputs the instruction code necessary to

manufacture the part on a computer numerically-controlled (CNC) machine

[Westhoven, 1991]. Its input is the feature-based description of the product from the

FBDE which is translated into fabrication features. FAB-PLAN uses a machining

database called Met-CAPP™, which determines the proper milling machine and the

corresponding speeds and feeds for each fabrication feature, then passes them as

parameters into the process planner.

*Inspection Planning and Evaluation Module* (IPEM) automatically generates instruction code to be executed on a computer-controlled coordinate measurement machine (CMM) [Merat and Radack, 1992] [Merat, et al., 1991]. IPEM is the module that this thesis discusses. The first step is the translation of the design/tolerance feature combinations from the FBDE into inspection features. The inspection process planner uses a generative approach to create the setups and sequencing of the tolerance measurements and evaluations. The path planner ensures an efficient path trajectory according to an inspection rule criteria, and then makes it a collision-free path. The automated code generator translates the inspection features into a language that the CMM will understand.

## 2.2. The Problem

As computers become faster, heuristics more comprehensive, and software more user-friendly, many new technologies have been realized in the industry arena. CAD (computer aided design) and CIM (computer integrated manufacturing) have become standard tools for most manufacturing companies. On a much smaller scale, CAI (computer aided inspection) and CAPP (computer aided process planning) have also been accepted by industry. However, these advances have resulted in distinct "pockets" of computer-assistance which are not complete in themselves, and integration among them is still labor intensive.

Many tools have been developed to aid the inspector with the inspection process. The tolerance standard, ANSI Y14.5M - 1982, provides a tolerance language that communicates to the inspector the intended function of the product. Computer-aided inspection planners prepare the detailed work instructions to inspect a part.

Stand-alone computer programs are also available to the inspector that aid in auto-mated inspection code programming. One type of program offers a graphical user in-terface that creates a plan by the concatenation of tolerance macros linked to graphi-cal buttons. Another software package will input an engineering drawing by a stan-dard interface, tracing, or some other manual technique, and simulate the probe path by overlaying the connected specified measurement points over the part-model.

While these computer aids are useful, considerable human interface time is re-quired to use the products together, not to mention separately. There are so many different types of information standards that these computer products are not connect-able, and translating between them is often more work than the services provided. For example, many CAD systems cannot represent Geometric Dimensioning and Tolerancing (GD&T) properly and may not even support it at all. Process planners need both the tolerances and a 3-D representation of the part model including surfaces and features: many CAD systems can only use 2-D geometric primitives. Most process planners output a text file containing manual instructions describing the sequence of tolerance measurements and setup orientations. However, this information cannot be used directly by the automated inspection software generator; therefore, the inspector must translate the manual instructions into automated CMM code.

## Motivation

The motivation for this thesis is to relieve the inspector from the tedious and time-consuming chores of inspection planning which can be automated by the com-puter. To reveal the needs of the inspector, the daily operations of the QC engineers at the 4950th Test Wing were observed. First, a manufactured product and

engineering drawing is given to them. Hours are spent to interpret and understand
the design function of the product as a whole, and the features within the product. If
the product is not toleranced using the GD&T standard, then datums and datum
reference frames (DRFs) must be created. An inspection strategy is created which
specifies the setups, fixturing, and DRF sequencing into the setups. The most tedious
and error-prone task is next: the placement and sequencing of inspection points into
a collision free path. This process includes applying 3-D trigonometry with
accuracies of thousandths of an inch. Currently, the inspectors use a hand-held
calculator to perform these calculations. Finally, the automated inspection code is
created by typing the coordinate points, tolerance values, and cryptic, hard-to-
remember CMM instructions into a simple word processor. This process is extremely
tedious and error-prone, yet critical, since any typographical error could send the
probe crashing into the product or CMM table. If the computer can be employed to
perform these routine but complex tasks, then the inspector can focus his expertise on
unique and unexpected problems that no algorithm can be programmed to foresee.

## Constraints

The constraints of the RDS system for the work discussed in this thesis are: (1)
prismatic products, (2) products made from aluminum stock, and (3) execution of the
inspection plan on a CMM with three degrees of freedom.

## 2.3. Previous Work

There are several bodies of literature relevant to this work: process planning,
optimization, collision avoidance, inspection techniques, code generation, and evalu-

ation methods. Although there is a significant amount of material in all these areas, there is little integration among them. The focus of this work is to combine and integrate these areas. Some of this work is paralleled in automated process planning for manufacturing. Caroline Hayes (1990) gives many references to previous machining planning systems.

Most of automated inspection planning literature has been described in the academic arena. ElMaraghy and Gu (1988) have described an inspection task planning system for CMMs. Their system is based on a feature-oriented computer-aided modeling system which was limited to cylindrical starting stock and turned parts. One significant aspect of their work was the use of the ANSI Y14.5 tolerances to govern their rule-based system. Other similar research has been done in the field of vision-based inspection systems. Traband and Medeiros (1988) describe a methodology for extracting the design information from a CAD system to control a two-dimensional video inspection system.

An important part of the inspection plan is the placement of the inspection points. The placement of the points to be measured must be accessible to the CMM probe within the given setup. If the point is not accessible, the setup must be changed, or probe extensions and other degrees of freedom of the CMM must be utilized. Spyridi and Requicha (1990) discuss using accessibility cones to determine if the inspection points can be reached by the CMM probe.

Hopp and Lau (1985) present two alternative control systems for generating an inspection plan and then creating the CMM commands to execute the plan. The first system uses feature decomposition on functional and tolerance features. The second system uses geometric decomposition to translate surfaces directly into inspection points.

The IPPEX, a knowledge-based system for dimensional inspection, performs its inspection task based on pre-processed, solid-model geometric information as well as tolerance information [Brown, 1990 & 1983]. The output is in the form of DMIS (Dimensional Measuring Interface Specification) code to operate a CMM. The major accomplishment of the IPPEX was an expert system to determine which type of CMM would be the best to perform an inspection plan based on the part size and inspection plan complexity.

Menq et al. (1991a, 1991b, 1992a, & 1992b) have developed an intelligent planning environment for automated dimensional inspection using CMMs. Their system is limited to parts described by complex and sculptured surfaces within the IBM CATIA CAD/CAM system. They excelled in developing localization algorithms to mathematically locate the part on the CMM table prior to inspection using complex surface fitting. However, this method does not use the ANSI Y14.5 method of creating a datum reference frame and then comparing the measurements to the theoretically perfect coordinate axes it creates.

The work presented by Jeon (1990) used an artificial neural network to sequence the inspection points using a Euclidean distance weight criterion. To meet inspection methodology standards, the sequencing was limited to a per surface basis.

One interesting industry application reported using the Hopfield artificial neural network as a discrete event sequencing problem in the area of hot strip milling for steel production [Kosiba, et al., 1992]. The neural network used a penalty function based on steel width, hardness, and gauge as constraints to search for the minimum cost solution. The result was a minimal cost path that defined the sequence of steel orders that make up the batch runs.

## 2.4. Object Oriented Programming

Object-oriented programming (OOP) is a method of software implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are members of a hierarchy of classes united via inheritance relationships [Booch, 1991]. Unlike conventional programming which is built upon algorithms, OOP is built upon objects. These objects are created from instantiations of classes. Classes are defined in an inheritance structure, where a sub-class can inherit properties from its super-class. At the very top of the inheritance chain is usually a class called a primitive, which is a pre-defined class that is a standard with the particular software implementation.

As a simple example, consider an object instance called game-ball that represents a soccer ball used for games only. This object instance can be associated with (instantiated from) sub-class soccer-ball which has properties size, stitching, and color. Other instances of this same sub-class soccer-ball could be practice-ball or loaner-ball. The differentiating features among the instances are the values of their properties (or slots) when the object instance was created. The sub-class soccer-ball would inherit attributes from its super-class ball, having properties air-inflation-pressure and material. The class ball is defined from the primitive class sphere with property radius.
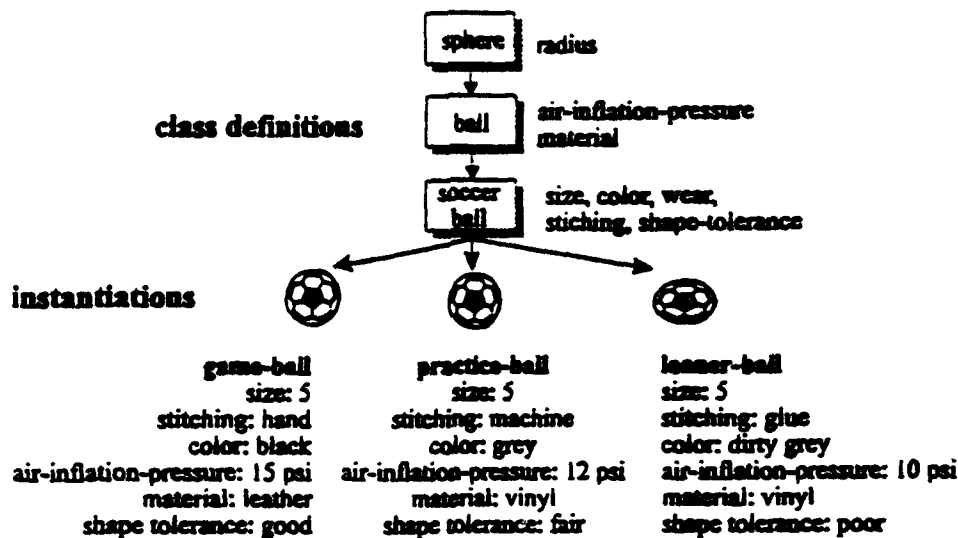
**Figure 4.** Class hierarchy illustration.

## 2.5. Features

The quest for completely automated process planning systems has exposed the lack of techniques capable of automatically understanding the stored CAD models in a manner suitable for process planning [Joshi and Chang, 1990]. Traditional CAD systems enhance product design simply by automating the drafting process. They operate at a low level in terms of product representation, i.e., the design information is represented in terms of geometric and topological primitives: points, lines, surfaces, etc. The part description in a 3-D CAD model (parametric surfaces and boundary representation) is in a form unsuitable for direct application to process planning. Therefore, many process planners have to interpret the CAD information using a feature-recognition pre-processor to convert it to meaningful manufacturing or inspection process information.

The approach of this research is to eliminate the ambiguities between computer assistance programs that have required the use of the manual translation or automatic pre-processing before the process planner can be utilized. The use of features within the CAD system and process planning or CIM/CAI can provide a common language among them. The features can propagate not only geometry, but also design intent, functionality, and other part expectancies from the designer to the manufacturer and inspector. Integrating the process planner to CAD using the feature paradigm involves a feature-translator that uses rules to map the features through the interfaces.

A feature has been defined in many ways. One definition is "computer representable data relating to functional requirements, manufacturing [or inspection] process, or physical properties of design" [Joshi and Chang, 1990]. The most common features, representing geometrical design, include through holes, blind holes, edge cuts, pockets, ribs, etc. (Figure 2 on page 7). Negative features are defined to consist of negative volume, or they take away material from their attachment feature or sub-feature. A hole is an example of a negative feature. Positive features are defined to consist of positive volume, or they add material to their attachment feature or sub-feature. A rib is an example of a negative feature.

Another type of feature represents the GD&T tolerances which are attached to other features and surfaces (Figure 3 on page 8). The combinations of design and tolerance features create inspection features which represent the finite elements of the inspection plan.

## 2.6. Concurrent Engineering

Concurrent engineering (sometimes called simultaneous engineering or life cycle engineering) involves the simultaneous consideration of product, function, design, materials, manufacturing processes, and cost, taking into account later-stage considerations such as testability, serviceability, quality, reliability, and redesign [Young, et al., 1992]. Concurrent engineering involves the consideration of all aspects of proper creation and life-duration of the product as early as possible — the design stage. This is especially important in small batch manufacturing and quality control operations since it is at the design stage that the life cycle requirements are defined. The decisions of the designer affect every aspect of the successful creation of that product.

The United States has a reputation that *concurrent engineering* techniques are generally not well performed [Young, et al., 1992]. It has been suggested (with tongue in cheek) that the designer "throws the plans over the wall" to the manufacturer who changes much of the design's tolerances when it is discovered that they cannot be met. The manufacturer then "throws the part and the designs over the wall" to the inspector who discovers that many of the tolerances were not met, and the part must be reworked. The result is an unnecessary increase of work, cost, and time accumulating into the product.

There are a number of techniques and systems that support concurrent engineering by advising designers on aspects that reduce life cycle problems. These include design teams, design handbooks, checklists and structured procedures, manu-

facturing (and inspection) simulation and process planning, and the use of expert systems [Young, et al., 1992].

In the RDS, concurrent engineering is accomplished through the use of features as the keys to the expert systems. The features have knowledge about themselves, i.e., they have rules on how they are to be manufactured, inspected, and interrelate with one another. Therefore, rules and constraints of the manufacturing and inspection disciplines can be attached to the design and tolerance features in the FBDE to provide the designer with knowledge that would not normally be known. For example, a hole feature is placed onto the starting block by the designer. The designer will be warned that a constraint has been violated if he makes the height:radius ratio too large, which causes the machining drill to chatter. The solution is a wider, shorter, or tapered hole. Accordingly, if a datum reference frame is created with the tertiary surface having much more surface area than the secondary surface, the designer will be warned that the opposite situation is desirable for optimum inspection.

Another aspect of concurrent engineering that is not covered in this research is the use of the EAM. Problems and solutions encountered in the manufacturing or inspection arena are stored with the part model at the time of occurrence. Then, at a later date when a similar part is introduced into the RDS by the designer, that wealth of past experience is available to the designer.

# Chapter 3

# Quality Assurance

*I don't have to be what you want me to be.*
*- Muhammad Ali*

The importance of quality control has been heightened in recent times by the increasing precision of manufacturing. The traditional approaches to dimensional inspection have become the bottleneck of the production line. [Menq, et al., 1992a]. As a result quality control has evolved from a trade to become its own field of study and has received recognition as a separate discipline within the science and engineering communities. New dimensioning and tolerance schemes, through the application of ANSI standards, have defined a more universal method of defining and communicating engineering intent [Brown, 1983]. Measurement methods have evolved from manual functional gaging into highly sensitive probing sensors and magnification optical systems. These technologies have now progressed from manually driven procedures into automated probing robots and vision systems with feature recognition.

## 3.1. Geometric Dimensioning and Tolerancing

GD&T is a means of dimensioning and tolerancing a drawing with respect to the actual function or relationship of part features which can be produced most eco-

nomically [Foster, 1986]. The key words are *function* and *relationship*. GD&T is a system of building blocks designed to make explicit tolerance requirements that otherwise would be interpreted only by implication. GD&T provides the designer with a clear way of expressing design intent and part requirements by providing a relationship between the toleranced feature and the datum features for the evaluation of the inspection measurements. This, in turn, allows the inspector to choose the proper coordinate frame in which to inspect the part, resulting in greater evaluation accuracy.

GD&T was created in the 1950s to avoid measurement ambiguity at its source — when the drawings are made and the tolerances are set. It was designed as a standard to help the inspectors understand and interpret the designers' meanings behind the tolerances so that the product can be properly inspected. This enhanced dialog among the designers, manufacturers, and inspectors was to help overcome the typical production procedure of "throwing the drawings over the wall" for the next group to try and interpret. GD&T can be considered in the same sense as a programmer providing comments as he develops software so that the software engineer who maintains the code (and other developers) has information in addition to just the code itself in order to understand what functions the code is supposed to perform.

This section will discuss the tolerances of GD&T, grouped together by the type of feature from which they are called. Table 1 shows the thirteen GD&T characteristics and symbols.

**Table 1.** Geometric Dimensioning and Tolerancing characteristics and symbols.

| Type of feature | Type of toler- ance | Characteristic | Symbol |
|---|---|---|---|
| Individual | Form | flatness | ⬭ |
| | | straightness | — |
| | | circularity | ○ |
| | | cylindricity | ⌀ |
| Individual or related | Profile | profile of a line | ⌒ |
| | | profile of a surface | ⌓ |
| Related (datum reference required) | Orientation | perpendicularity | ⊥ |
| | | angularity | ∠ |
| | | parallelism | ∥ |
| | Location | position | ⊕ |
| | | concentricity | ◎ |
| | Runout | circular runout | ↗ |
| | | total runout | ⟋ |

## Basic Dimensions

A basic dimension is a numerical value used to describe the theoretically exact size, profile, orientation, or location of a feature target. It has no tolerance placed on it, since it is from the basic dimensions that permissible variations are established throughout the part. They are identified by the word BASIC, the abbreviation BSC, or placed within a box.
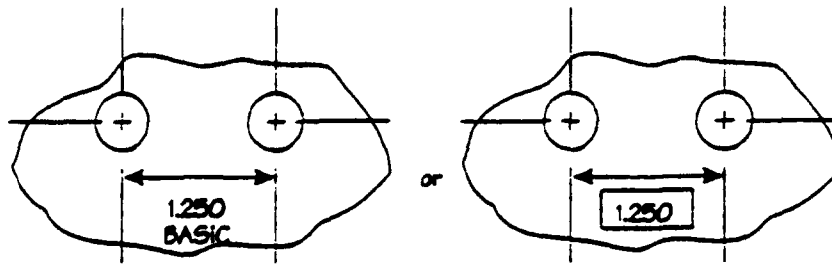
**Figure 5.** Basic dimension callout.

## Material Condition Modifiers

One of most important aspects of GD&T is the material condition modifiers. These modifiers can be used only on features of size such as holes, shafts, pins, and slots, as opposed to surface features. A modifier alters the tolerance zone of a tolerance callout depending on the feature's actual manufactured size versus its theoretical size:

The most common material condition modifier is the Maximum Material Condition (MMC, Ⓜ). This condition occurs when the feature of size has been manufactured at the largest or smallest allowable toleranced size which results in the maximum material stock remaining. Therefore, a hole at MMC will be the smallest allowable size within tolerance (minimum diameter), while a boss at MMC will be the largest allowable size within tolerance (maximum diameter). This principle permits a relaxed tolerance value (called "bonus" tolerance) as part feature sizes vary from the allowable MMC and still ensures proper feature functionality. At MMC, features are at their "tightest" tolerance: a hole is at its smallest, and a boss is at its largest. As these features drift from MMC — a hole gets a little larger and a boss gets a little smaller — there is more "play" at the location of the feature, so the toler-

ance zones are allowed to increase by the addition of the bonus tolerance, yet provide proper functional requirements. The amount of bonus tolerance awarded is equal to the distance that the feature drifts from MMC. This is one of the fundamental principles on which GD&T is based.

Figure 6 illustrates MMC and the bonus tolerance condition with a plate and its mating part. Figure 6(b) shows how the bonus tolerance of the hole that drifts from MMC allows the position of the hole to vary and still properly connect with its mating part.



(a) plate (top view)                    (b) plate (Section A-A)
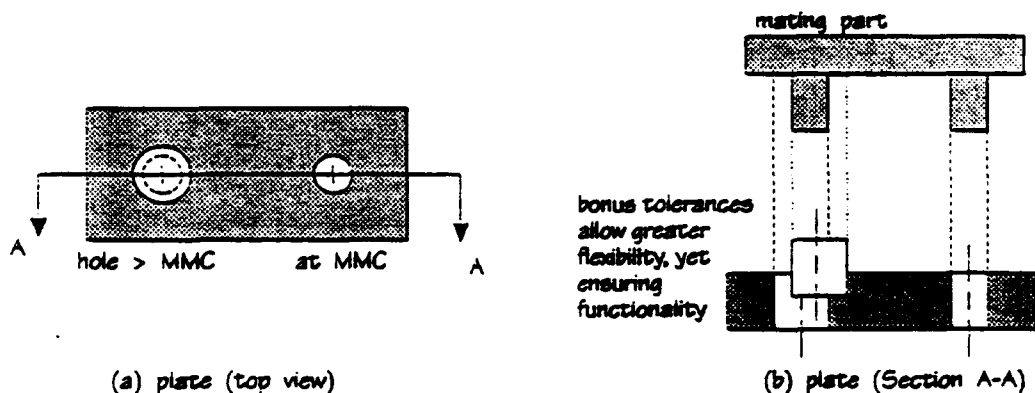
**Figure 6.** Bonus tolerances provide greater function flexibility.

MMC is one of three material condition modifiers. The others are Least Material Condition (LMC, ⓛ) and Regardless of Features Size (RFS, Ⓢ). LMC also awards bonus tolerance like MMC, only in the opposite fashion. If the feature of size drifts from the least amount of material stock under the allowable tolerance range, then the bonus is awarded. RFS does not allow any bonus tolerances, and the tolerance value is constant regardless of manufactured feature size.

## Datums

The tolerances in GD&T are referenced with respect to datums. A datum is a theoretically exact point, axis, or plane created from the true geometric measurements of the datum callout feature. The datums define the origin and coordinate reference frame axes from which the location or geometric characteristics of the features of a part are evaluated. Being theoretically exact means that all the surface and feature inaccuracies from machining, warping, etc. are inherent in the datum. This is why *relational* tolerancing is stressed so highly in GD&T. The tolerance evaluations are based on the their relationship to the datum features, e.g., what feature the part will rest upon, what axis the part will spin around, or what flange a pocket will mate with in assembly.
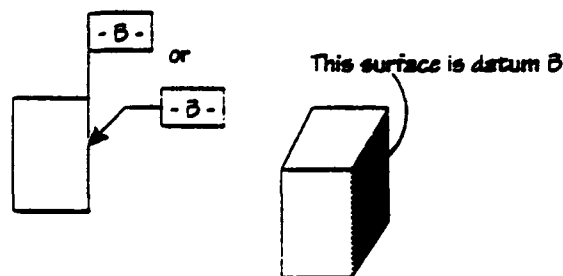


**Figure 7.** Datum callout.

## Datum Reference Frame

The datum reference frame (DRF) is composed of three mutually perpendicular datums. Selection of the datums is made by the functional importance of the features

and their relationship to the datums. How the product is constrained. whether fixed or in motion, during normal operation determines the datums.

For the datum planes to be in theoretically perfect normality in an imperfect manufacturing environment, the 3-2-1 convention is used. This refers to a mathematical formula which requires three points on the primary datum (usually the resting surface), two inspection points on the secondary datum, and one point on the tertiary datum (Figure 8). Since three non-collinear points are required to make a plane, each least significant datum uses points from the more significant datums to ensure perpendicularity. More technically defined, the 3-2-1 convention is a systematic method of constraining the degrees of freedom of the inspected object. The datums are typically surfaces but can also be axes of the part geometry.
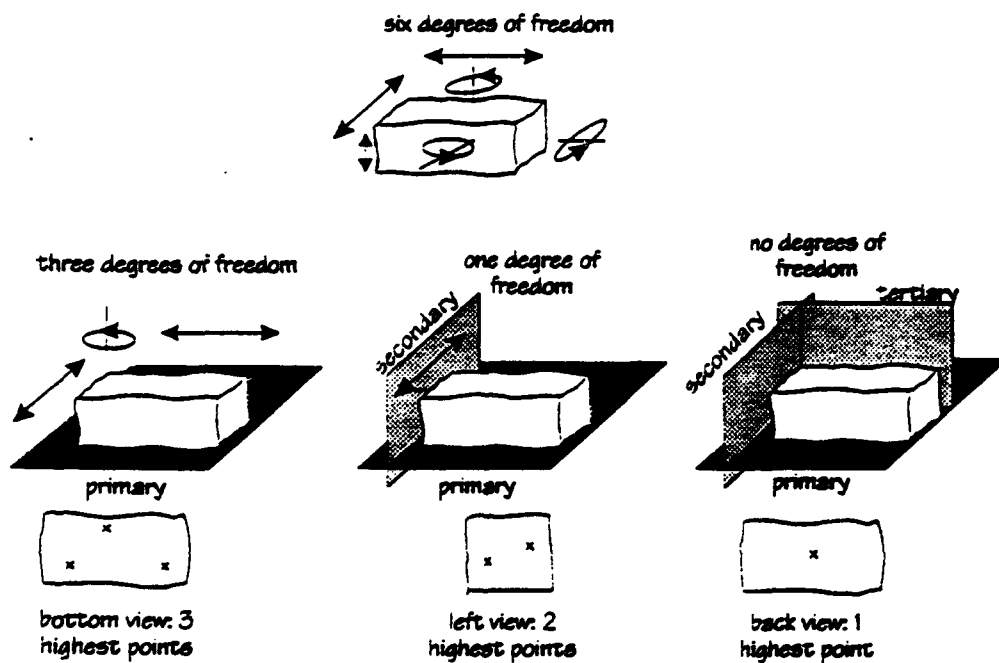


**Figure 8.** Creation of datum reference frame using 3-2-1 convention to remove all degrees of freedom from the part model.

## Independent Features

An independent feature is a single surface, element, or size feature which relates to a perfect geometric counterpart, or theoretically perfect copy, of itself as the desired form. There are no datum references used with these tolerances, which include straightness (⌐), flatness (▱), circularity (O), and cylindricity (⌀).

*Straightness* is a condition in which an element of a surface or an axis is in a straight line. The straightness tolerance defines two different tolerance zones, depending on the feature called from. Surface straightness defines two parallel lines, distanced apart by the tolerance value, that the surface element must lie between. Axis straightness defines a cylindrical tolerance zone for the axis to lie within.
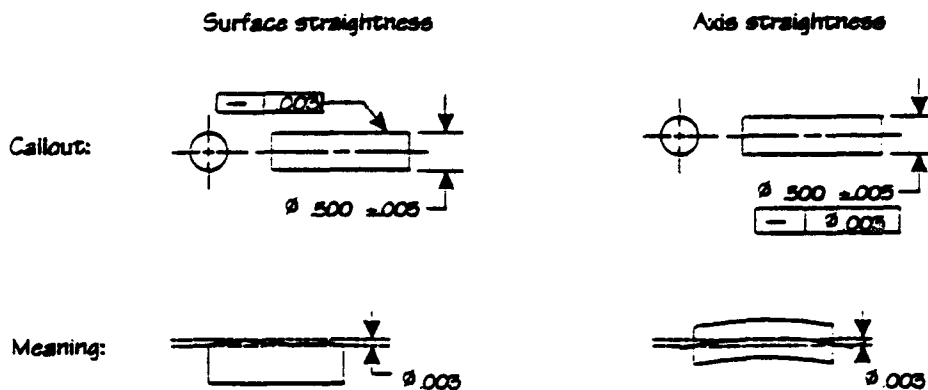


**Figure 9.** Straightness tolerance callout and meaning.

*Flatness* is the condition of a surface having all elements in one plane. The flatness tolerance specifies a tolerance zone confined by two parallel planes within which the entire surface must lie. The tolerance may also be used for a specified area of a surface, rather than the entire surface.
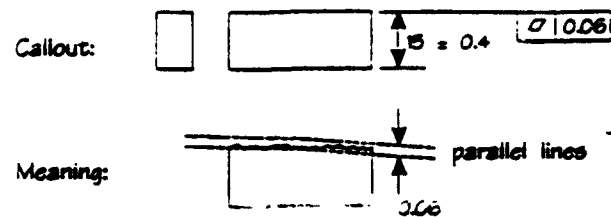
Callout:

Meaning:

Figure 10. Flatness tolerance callout and meaning.

*Circularity* is the condition on a surface of revolution where all points of any given cross section, taken perpendicular to the axis of a cylinder or cone or through the common center of a sphere, are equidistant from that center. The tolerance zone is bounded by two concentric circles within which the actual surface must lie.
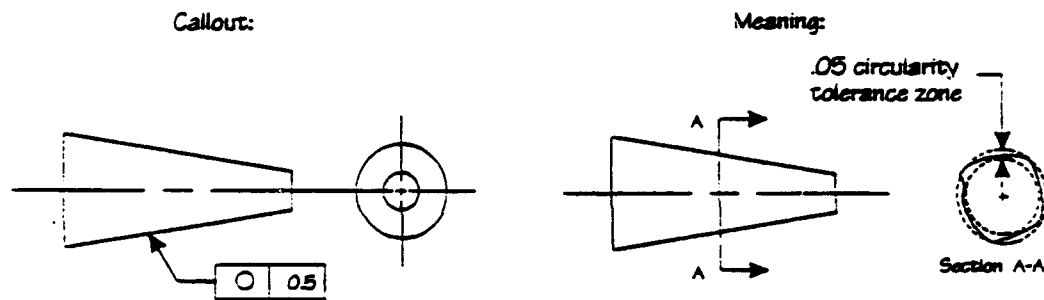
Callout:                                    Meaning:

Figure 11. Circularity tolerance callout and meaning.

*Cylindricity* is the condition of a surface of revolution in which all points of the surface are equidistant from a common axis. The tolerance zone is bounded by two concentric cylinders within which the toleranced surface must lie. The cylindricity tolerance simultaneously controls circularity, straightness, and parallelism of the elements of the cylindrical surface, since it covers both circular and longitudinal elements of the toleranced surface at the same time.
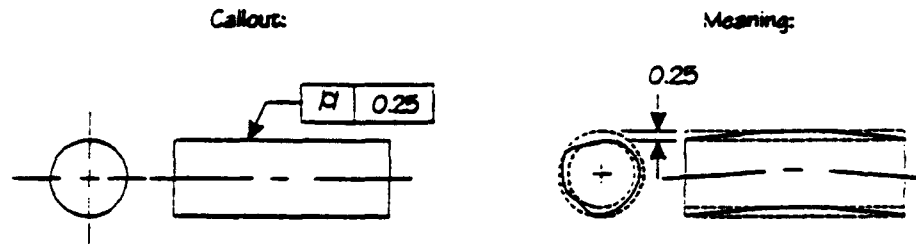
**Figure 12.** Cylindricity tolerance callout and meaning.

## Related features

A related feature is a single surface, element, or size feature which relates to a datum, or datums, in form and/or orientation. A datum or datum reference frame must be included in the feature control frame. Related orientation tolerances include perpendicularity (⊥), parallelism (//), angularity (∠). Related location tolerances include position (⊕) and concentricity (⊙).

*Perpendicularity* (also called squareness or normality) is the condition of a surface, median plane, or axis which forms exactly a 90° angle to a datum plane or axis. The tolerance zone for a toleranced surface is created by two parallel surfaces that are perpendicular to the datum surface within which the tolerance surface or median plane must lie. The tolerance zone for a toleranced axis is created by a cylinder perpendicular to the datum plane within which the toleranced axis must lie.
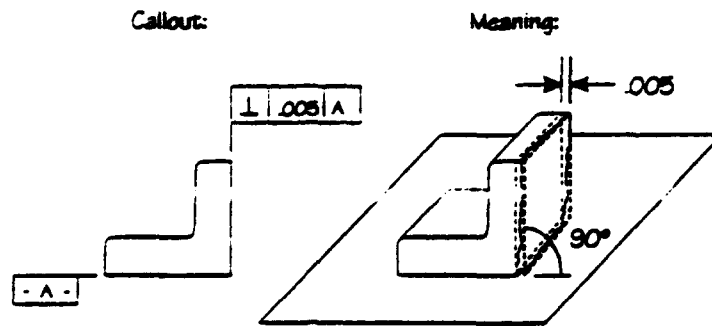
Callout:                          Meaning:

Figure 13. Perpendicularity tolerance (surface) callout and meaning.

*Parallelism* is the condition of a surface or axis which is equidistant at all points from a datum plane or axis. When a surface is toleranced, the tolerance zone is defined by two planes parallel to the datum plane between which the toleranced plane must lie. When an axis is toleranced, the tolerance zone is defined by a cylindrical tolerance parallel to the datum axis within which the toleranced axis must lie.
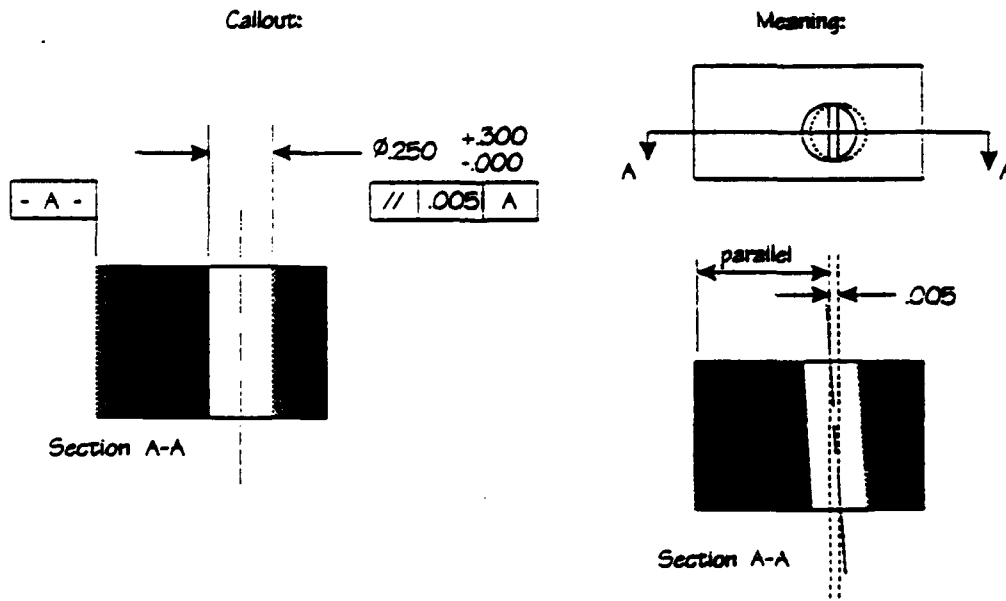
Callout:                          Meaning:

Figure 14. Parallel tolerance callout and meaning.

*Angularity* is the condition of a surface, axis, or median plane which forms a specified angle (other than 90°) from the datum feature. The tolerance zone is created by two parallel planes distanced by the tolerance value, inclined at the specified angle from the datum plane or axis, and within which the toleranced feature must lie.
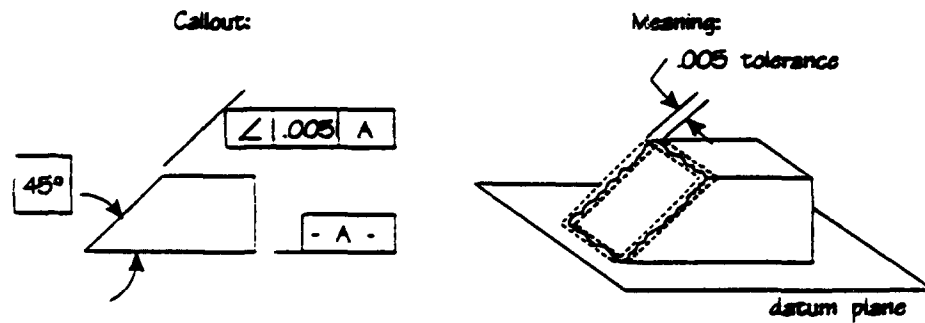


**Figure 15.** Angularity tolerance callout and meaning.

Location tolerances state the permissible variation in the specified location of a feature in relation to some other feature or datum. They define a zone within which a center, axis, or center plane of a feature is permitted to vary from true position. Location tolerances include position (⬥) and cylindricity (**O**).

*True Position* describes the exact location of a point, line, or plane of a feature in relationship to a datum reference or other feature. The position tolerance zone is the total permissible variation in the location of a feature from its true position. For cylindrical features (holes and bosses), the tolerance zone is a cylinder whose diameter is the tolerance value and within which the axis must lie. For other features, i.e., slots, pockets, etc., the tolerance zone is two parallel planes separated by the true position tolerance value, within which the surface or center plane must lie.

In comparison to the conventional ± tolerance method, GD&T's use of position tolerancing provides some great advantages. First, the tolerance zones are measured

from the datums that are functionally relevant, not from a convenient origin as with many coordinate tolerance drawings. These datums are represented in the Datum Reference Frame (DRF), which determines the coordinate axes frame in which the part is to be measured based on how the designer intends the part to be operated or assembled. Second, the tolerance zone is more accurate and actually larger. For a toleranced axis, the tolerance zone is cylindrical, not square; this provides an area increase of 57%, and that is without any bonus tolerances!

$$\sqrt{2} \cdot x = 1.4 \cdot x$$

bonus tolerance
(if applicable)

$$\pm \frac{x}{2}$$
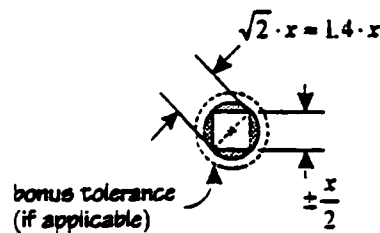
**Figure 16.** GD&T provides a greater tolerance area with a circular tolerance zone.

When a material condition modifier is placed in the feature control frame of the position tolerance, feature size and location are interdependent. Bonus tolerance will be added when the feature's size varies from the specified material condition modifier.

HOLE          MMC                    LMC

bonus tolerances

BOSS
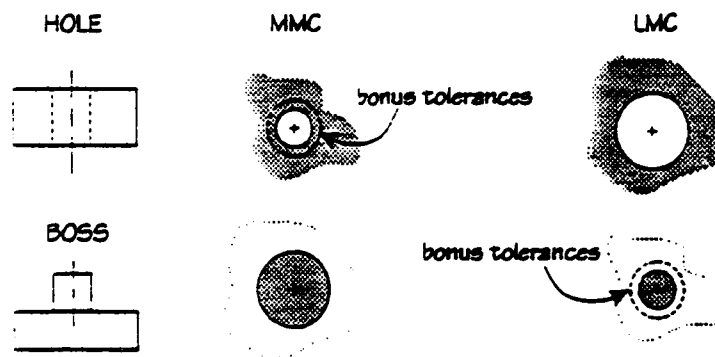
bonus tolerances

**Figure 17.** Bonus tolerances applied to the material condition modifiers.

*Concentricity* is the condition in which the axis of all cross-sectional elements of a feature's surface of revolution are common to the axis of a datum feature. The tolerance zone is a cylinder with the diameter of the tolerance value within which the axis must lie.
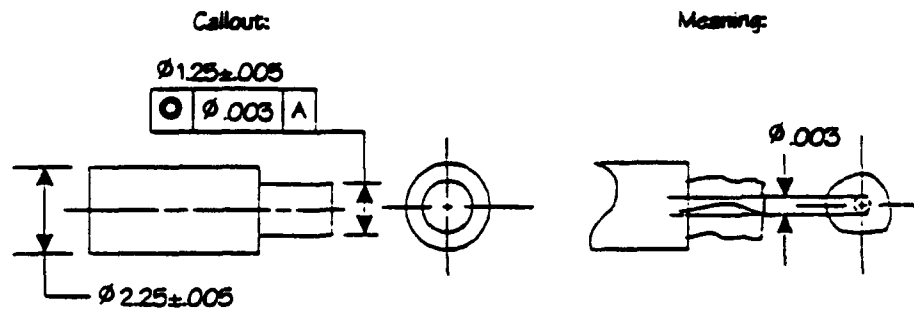


**Figure 18.** Concentricity tolerance callout and meaning.

## 3.2. Computer-controlled Inspection

A number of different technologies have been developed to automate the inspection process. The most popular have been vision systems and coordinate measurement machines (CMMs) which are beginning to be used extensively in industry for automatic dimensional inspection of products [Etesami and Qiao, 1989]. The CMM has become very popular due to new technologies increasing its speed and accuracy, allowing more than 60 measurements per minute with accuracies to 0.00001 inch. Therefore, the new challenges surrounding CMMs are not seen in the act of data retrieval, but rather the planning before and after data acquisition and retrieval — where to take the inspection measurements, how to sequence the inspections for efficiency, and how to evaluate them once they are taken.

A CMM can be regarded as a Cartesian robot whose end-effector is a contact probing system (Figure 19). Its task is to perform dimensional measuring by touching the surfaces of a located workpiece and reporting an accurate location based on a predetermined origin. The original CMMs were manually operated, displaying one measurement at a time. Today's CMMs are computer-controlled and driven by programs written with software languages that represent an entire workpiece inspection. Many of the computer controllers will also evaluate the measurement data and answer "yes/no" as to whether the part is within tolerance.

As is the case with any new technology, there are both advantages and disadvantages. CMMs have produced time and labor savings by replacing some classical approaches such as open set-up and hand-tool dimensional inspection techniques which are costly, inherently slow, less accurate, and subject to inspector error. The primary benefits of the CMM are its reduced setup time, greater accuracy, dependability and repeatability, and automatic operation. However, CMMs are initially very expensive. Even after a significant amount of use, CMMs require a skilled programmer with a quantity of time allocated for each job. A shop with many small lots may not see a time benefit from the CMM due to the programming and process planning time required.

coordinate measurment machine (CMM)
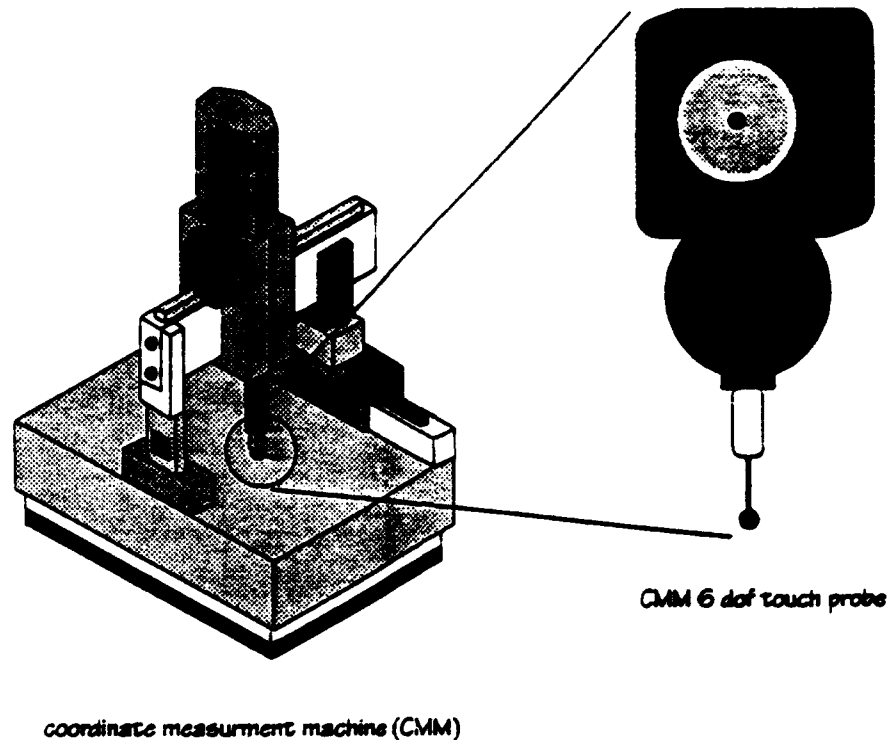
**Figure 19.** Coordinate measurement machine.

## 3.3. Automated Inspection

The inspection tools mentioned above, GD&T and CMMs, have enhanced the ability of the Quality Assurance engineer to inspect manufactured products. However, the tools have introduced their own problems that must be solved. First, a new dimension has been added to creating the process plan. Not only must the tolerances be measured correctly, but measurement instructions must be translated into CMM instruction code to automatically inspect the product efficiently and safely.

Second, the two tools are not collaborating, i.e., one is not complementary to the other. GD&T is based on hard gaging, a technique that uses surface plates, plugs,

gage pins, perpendiculars, functional gages, etc. CMMs do not measure in the same way as these hard gages do, and therefore produce different results. CMMs measure by single-point measurements and a mathematical averaging to create virtual features. For example, a CMM will measure three points on a datum surface. A mathematical algorithm will take those three points and construct the normal for a virtual plane which will represent the actual datum from which all other measurements will be compared. Unless the three measured points were the highest three points on the actual surface, the virtual plane is going to be incorrect both in position and orientation. A surface plate is guaranteed to pick up the highest three points that will become the datum.

Two approaches can be applied to overcome this deficiency and "marry" the two techniques together. In the first approach, the CMM is used to meet hard gaging requirements of GD&T. For example, the inspector would use a gage pin or surface plate, and then measure the inspection points from the gage using the CMM. However, the extensive user time involved would defeat the speed of the computer-controlled CMM, and other cheaper machines could be used.

In the second approach, the GD&T standard is enhanced so that the benefits of the computer-controlled CMM can be realized. A Y14 Ad Hoc Mathematization Committee and the ANSI/ASME B89.3.2 committee have been established. These committees are working on "mathematizing" the current Y14.5 standard [Schreiber, 1990]. This task includes establishing unambiguous mathematical definitions of tolerancing and sampling procedures based on features and processes.

To illustrate, the diameter of a hole feature should be measured differently than the diameter of a boss feature because each feature has a different function. For a hole, the minimum inscribed radius is desired because a mating boss has to fit into it

and can only be as large as the minimal radial separation. Using a hard gage will automatically provide this value. For a boss, the maximum circumscribed radius is desired. However, most CMMs now use a least squares averaging algorithm for both features, which can even be viewed as functionally incorrect.

Evaluation Method



Maximum Inscribed: hole

Minimum Circumscribed: boss

Least Squares: CMMs
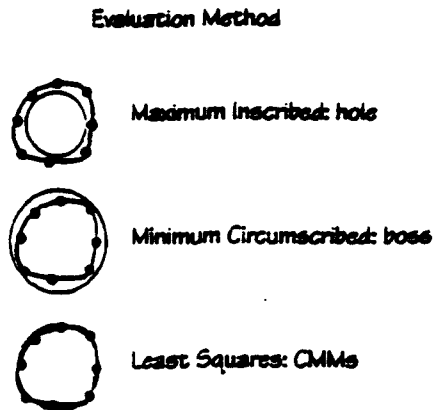
**Figure 20.** Three different evaluation results based on the same inspection points.

# Chapter 4

# Inspection Planning and Evaluation Module

> *Nothing is so commonplace*
> *as to wish to be remarkable*
>       - Oliver Wendell Holmes

---

Process planning is defined as the act of preparing a detailed plan for the production (i.e., manufacturing, inspection) of a part or assembly [Brown, 1983]. Process planning requires a significant amount of both time and experience. According to an Air Force study, a typical process planner is a person over 40 years of age with significant experience in a machine shop [Chang and Wysk, 1985]. Most experts agree that process planning is not an exact science, but is more of an art gained from years of experience. The automation of process planning would provide significant aid to the job shop. With automated process planning, the learning curve of new engineers/technicians could be shortened and their productivity could be increased. The experienced process planners could focus their attention onto the unique problems that arise, allowing the automated system to plan for the routine recurrent operations. According to Chang and Wysk (1985), the advantages of computer-aided process planning are:

- It can reduce the skill required of a planner.
- It can reduce the process planning time.

- It can reduce both process planning and [inspection] costs.
- It can create more consistent plans.
- It can produce more accurate plans.
- It can increase productivity.

A parallel to inspection, automated manufacturing planning systems have received much attention from research and industry over the past 20 years [Westhoven, 1991]. Hayes (1990) names 22 automated manufacturing process planners. However, automated process planning for inspection is relatively new [Traband and Medeiros, 1988]. Brown (1983) describes three automated inspection systems and also states that "[d]ocumented efforts toward automating computer aided process planning for inspection have been few, especially in comparison to the CAPP efforts directed toward manufacturing the part."

An inspection process plan would include the detailed sequence of events to successfully satisfy inspection evaluation. These events include:

- describe part orientations to access inspection points
- locate part on inspection table (create part-model coordinate frame)
- inspect and create DRF coordinate frame
- inspect feature tolerances using proper methods for tolerance callout
- evaluate measurements
- report results to the human inspector
- prepare for the next measurement in an efficient and safe manner

The events can be in the form of printed instructions for manual inspection, or the form of code that can be executed by automated inspection equipment to perform the inspection plan [Merat, et al., 1991].

## IPEM Overview

The IPEM can be described by two parts: the process planner and the CMM code generator. The inspection planner translates the design/tolerance feature combinations and part-model geometry into a suitable form for the code generator. Section

4.1 describes the work of the IPEM research team on the process planner. Sections 4.2 and 4.3 describe the individual work accomplished that created this thesis: the automated CMM code generator. This includes three main efforts:

1. an artificial neural network to ensure an intelligent scheduling of measurement points based on a set of inspection rule criteria (section 4.2.2),

2. a collision avoidance algorithm using computationally efficient path-generation methods (section 4.3.2), and

3. the inspection plan translation into CMM code, structured into a format the inspector can understand and interpret easily (section 4.3.4).

Figure 21 diagrams the algorithm of the IPEM. The boxes with the black shadow, rather than gray, are the areas of individual effort accomplished for this thesis.
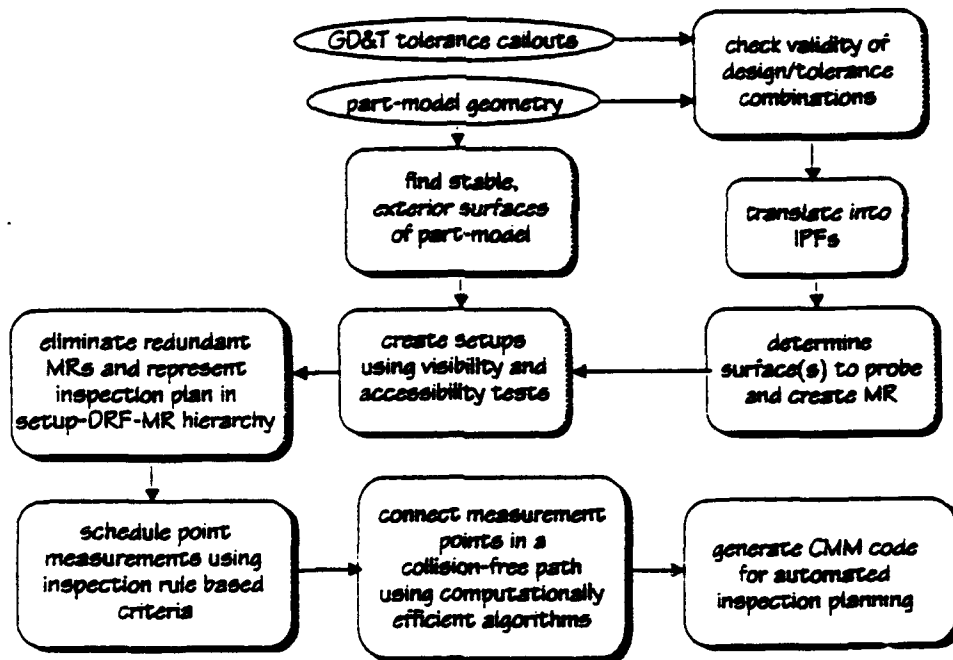
Figure 21. IPEM algorithm.

## 4.1. Automated Inspection Process Planning

The function of the automated process planner in the IPEM is to input the design features, consisting of geometry and tolerance features, and to output an inspection plan representation which serves as a template for the final inspection plan form — CMM code. The output plan is defined by a set of hierarchical structures consisting of setup/datum reference frame/tolerance measurement objects. This structure characterizes the GD&T standard sequence of events to take place in the inspection process. This sequence includes four main operations:

- orient and locate part in proper setup
- create virtual coordinate frame
- measure tolerances
- evaluate measurements

The process of creating this sequence of events to inspect the workpiece is called inspection plan sequencing throughout this work.

### 4.1.1. CAD Feature Translation

This thesis has emphasized the importance of features used not only as geometry representation, but also as the common link between the different engineering modules. Within the FBDE, new tolerance features are checked by constraint managers for appropriateness as defined by GD&T convention. For example, a circularity tolerance feature cannot be placed on a pocket design feature. A constraint violation is defined consisting of "Y" for appropriate, "NS" for non-standard but acceptable, and "NA" for not appropriate. Presently, the designer can override the constraint violations and place any tolerance on any design feature.

Table 2. Allowable design/tolerance feature combinations within the IPEM.

| Design → <br><br> Tolerance ↓ | Blind / Through Hole | Boss | Pocket | Through Slot | Open Step | Edge Cut | Rib | Surface |
|---|---|---|---|---|---|---|---|---|
| Flatness ▱ | NA | NA | NA | NA | NA | NA | NA | Y |
| Straightness — | NA | NA | NA | NA | NA | NA | NA | Y |
| Circularity ○ | Y | Y | NA | NA | NA | NA | NA | NA |
| Cylindricity ◎ | Y | Y | NA | NA | NA | NA | NA | Y |
| Perpendicularity ⊥ | Y | Y | NS | Y | Y | Y | NS | Y |
| Angularity ∠ | Y | Y | NS | Y | Y | Y | NS | Y |
| Parallelism // | Y | Y | NS | Y | Y | Y | NS | Y |
| Position ⬦ | Y | Y | Y | Y | NA | NA | NS | NA |
| Concentricity ◎ | Y | Y | NA | NA | NA | NA | NA | NA |

The IPEM process planner begins by accessing design and tolerance information created in the FBDE and represented as features. The design/tolerance feature combinations are checked for current implementation. Since the IPEM is a research project, not everything is complete. Combinations that have not yet been implemented are simply noted to the user as incomplete.

Feature translation takes the combination of a tolerance feature and the design feature from which it is called and creates the inspection features. This first type of inspection feature is referred to as "inspection plan fragment" objects (IPFs). IPFs signify a segment of the total inspection plan. This translation is unique in that it is the only feature translation within the IPEM that has a 1:1 ratio, i.e., each design/tolerance combination is translated into exactly one IPF.

The next task is to determine what surfaces the design/tolerance feature combination requires to be measured. Since features have multiple surfaces, a different set of surfaces will need to be measured depending on the tolerance callout (Figure 22).

Inspection rules operate upon the IPF objects to specify the actual surface(s) to be measured. This operation creates a new inspection feature called the "measurement request" (MR).

**blind hole design feature**



surface to be
inspected

position tolerance callout                    flatness tolerance callout
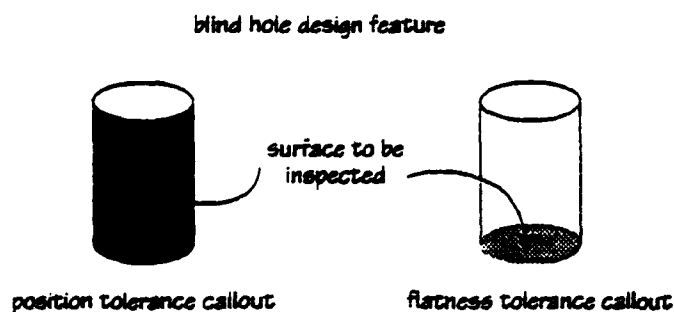
**Figure 22.** Each design/tolerance feature combination specifies a set of surfaces to be measured.

The translation of an IPF to an MR is not a 1:1 translation because there may be more than one tolerance per surface. When this occurs, the measurement points from one tolerance can often be used in a different mathematical interpretation to evaluate a different tolerance. For example, a datum plane could also have a flatness tolerance callout. The planar surface has two IPFs, i.e., two different tolerances (the datum and the flatness). The inspection plan needs only one measurement request but two evaluation requests since the same inspection points measured for the datum plan can be used for the flatness tolerance. This eliminates any redundant measurements of surfaces.

The MR is the feature class that will be extensively used by the IPEM for inspection process planning. This class is broken down into two types of MRs: independent ($MR_i$) and related ($MR_r$). $MR_i$s contain independent tolerances and $MR_r$s contain related tolerances. i.e., tolerances dependent upon datums or a DRF (section 3.1). Knowledge rules generate the inspection plan by sequencing, optimizing, and

translating the MRs based upon 3-D geometric reasoning, inspection rules, and AI heuristics.

## 4.1.2.  Structure Setup and Pre-plan Geometric Reasoning

Once the MRs are created from design and tolerance feature translations, the IPEM can begin to plan for the inspection plan sequencing. The inspection plan structure is represented by the sequence hierarchy of setup-DRF-MR. This notation refers to setup objects as parents of DRF objects, which are parents of MR objects. Using the CLOS class paradigm allows inheritance of properties from parent to child. For example, the inspection points of the child MR inherit the rotation matrix from a property within the setup parent which properly orients the points within that setup.
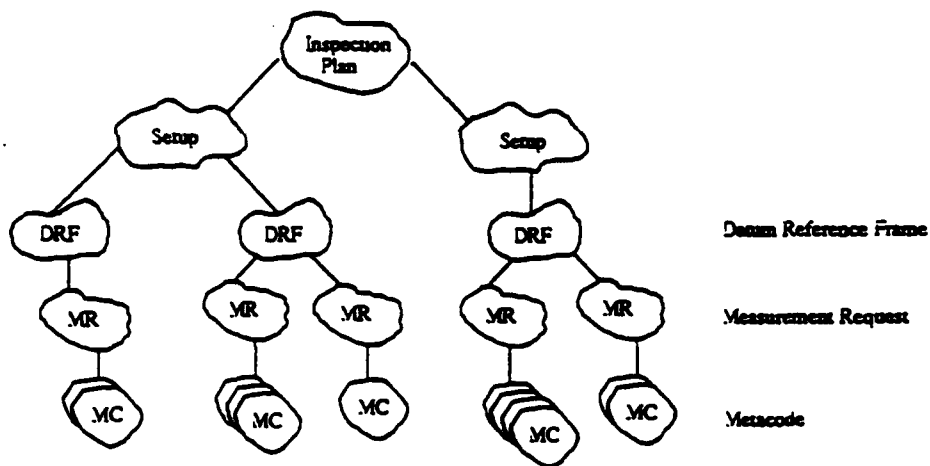
Figure 23.  Inspection plan hierarchy.

## The Setup Object

The setup object describes the orientation of the workpiece on the CMM table. It contains two important elements: the resting surface of the workpiece, and the orientation information, described by the angles needed to rotate the workpiece from the designer's original orientation used in the CAD system into the orientation of the setup. The angles create the rotation matrix, $R$. They are defined by a mathematical standard procedure for graphical rotations to place an object into an arbitrary orientation. The procedure is similar to Euler's angles or *roll-pitch-yaw* angles which condense the rotation matrix into independent variables only by removing redundant information. The rotation matrix is then defined by an algorithm performed on the independent variables. In the procedure used by this research, the rotation matrix is defined by:

$$R(\phi, \theta, \psi) = R_y(\phi)R_x(\theta)R_z(\psi) \tag{4.1}$$

where the individual rotations are:

$$R_y(\phi) = \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \tag{4.2}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & -s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix} \tag{4.3}$$

$$R_z(\psi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

with shorthand notations $c_x = \cos(x)$ and $s_x = \sin(x)$. Since matrix multiplication is not commutative, the matrices must be multiplied in the order of equation (4.1). Pre-defined functions make this easy to do in LISP. For example, to create a point, $x_2$, in the new setup coordinate frame with angles $y$, $x$, $z$, from the point, $x_1$, the LISP code would be:

```
(setf ry (calculate-rotation-matrix 0.0 y 0.0))
(setf rx (calculate-rotation-matrix x 0.0 0.0))
(setf rz (calculate-rotation-matrix 0.0 0.0 z))
(setf x2 (multiply-vector-and-matrix x1 ry))
(setf x2 (multiply-vector-and-matrix x2 rx))
(setf x2 (multiply-vector-and-matrix x2 rz))
```

Figure 24 shows how coordinate frame xyz is rotated into its setup orientation by three ordered rotations about the axes y, x', and z''. Each rotation represents one element of the R matrix.
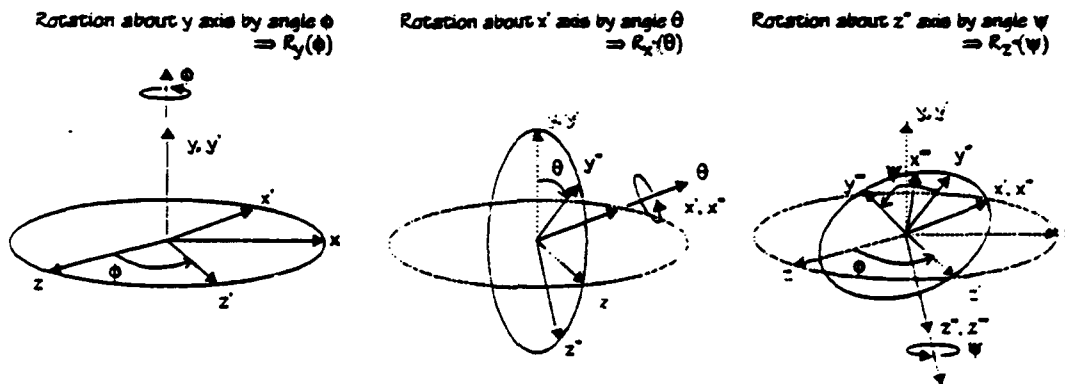


Figure 24. Three consecutive axes rotations.

## The Datum Reference Fame Object

The DRF object describes a virtual coordinate frame in its parent setup object. The coordinate frame is defined by the GD&T 3-2-1 convention of three mutually

perpendicular datums (Figure 8 on page 25). Therefore, the MRs of the three datums are contained in the DRF object. The DRF type is also stored as a property. DRF type is the concatenation of the three datum types, which refers to whether the datum is a plane or axis, represented as P or A, respectively. The most common DRF type is PPP, three planes. Another common type is PAP or PPA, where an axis and two planes constrain the part to zero degrees of freedom. This information is used by the output code generator to determine where the origin lies. If the DRF is of type PPP, then the origin is the intersection point of the three planes; otherwise, an algorithm must be used based on the position of the axis datum (section 4.3.2).
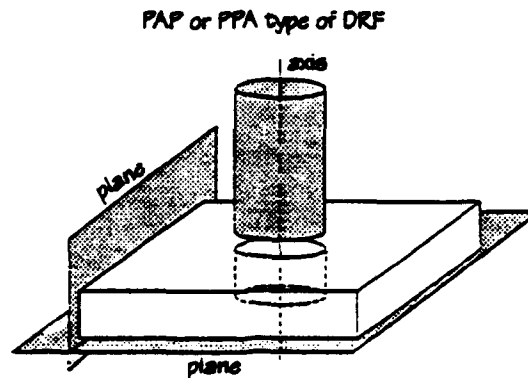


**Figure 25.** Datum creation within a DRF that contains an axis datum.

## The Measurement Request Object

As described above, the MR inspection feature is translated from the design and tolerance feature combinations of the FBDE via the IPF. Obviously, the properties of the MR contain the tolerance and feature types and the set of surfaces to be measured. However, before inspection plan sequencing can occur (determining the proper work-piece orientations and tolerance sequence within the inspection plan), more geometry

information is needed. First, inspection rules place measurement points on the toler-anced surfaces. The rules determine how many points and what constraints are re-quired to properly evaluate the design/tolerance feature combination of the MR (Table 3). For example, to create a datum, at least three non-collinear points must be measured.

---

**Table 3.** Inspection point constraints on placement based on design/tolerance feature combinations.

| Design feature | Tolerance feature | Number of points (min, default) | Constraints* (min, default) |
|---|---|---|---|
| (b/t) hole | perp. ⊥ | 3,4 | bi-plane, tri-plane-120 |
| | circ. O | 3,4 | planar, planar-120 |
| surface | perp. ⊥ | 3,5 | non-collinear |
| | primary datum | 3 | non-collinear, RHRup |

* Not all of the constraint options are implemented yet.

---

### Pre-plan Geometric Reasoning

The goal of the pre-plan geometric reasoning is to create all the allowable setup-DRF-MR object combinations, and then use AI heuristic searches to remove the redundant measurement structures. The approach is first to create the set of all possible setup objects, then to place the MRs into their proper DRF object, and finally to place the DRF-MR structure into each allowable setup object based on in-spection point visibility.

An extensive amount of geometric reasoning is used to prepare for setup object instantiation. The setup objects are instantiated from the stable surfaces of the part.

The stable surfaces are a sub-set of the exterior surfaces. The exterior surfaces are distinguished from all other types of surfaces of the solid part-model by two tests. The first test determines a surface is planar by checking for constant normals across the surface. Then offset surfaces are created on each side of the planar surface and "grown" within their plane to values greater than the known maximum dimensions of the part-model. If one of the offset surfaces does not intersect the part model, the original surface is determined as the external surface.
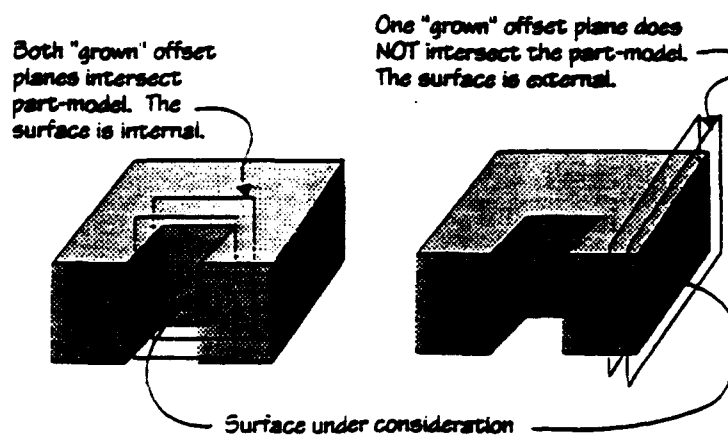


**Figure 26.** Growing offset surfaces to determine for internal or external status.

The external surfaces then are the candidates for resting surfaces of a possible part setup orientation. A stability test using a convex-hull algorithm [Preparata and Hong, 1977] selects which exterior surfaces can support the workpiece without the use of fixtures. Since fixtures require extra time and effort, make it hard to produce repeatability, and pose special threats in the form of probe collision problems, the stable resting surfaces are preferable to the inspector.

unstable resting surface    stable resting surface

convex hull area

projected center of mass
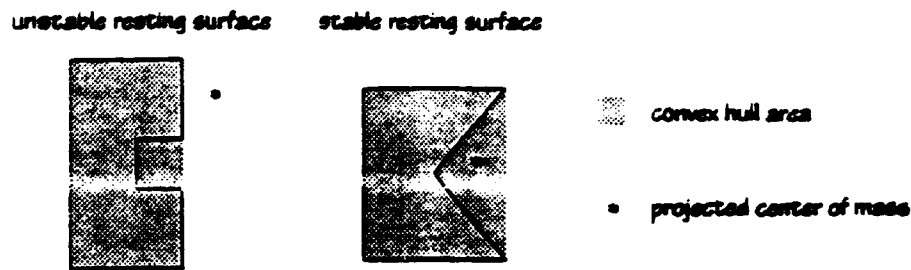
**Figure 27.** Convex hull results.

<u>Creating the Setup-DRF-MR Objects</u>

Each stable resting surface corresponds to one setup instantiation. The setup object is populated with the proper setup orientation matrix and stable resting surface.

Two GD&T guidelines are used to create the proper DRF-MR structures. Since the $MR_r$ (MR with related tolerance) is defined by GD&T to be inspected in its DRF coordinate frame, the $MR_r$ object is simply placed into the slot of the DRF specified from its related tolerance callout. However, the $MR_i$ can be measured in any coordinate frame; therefore, they are placed initially into every DRF.

The DRF-MR structures are now added to any appropriate setup object. The setup-DRF-MR structure is deemed appropriate by testing for inspectability, meaning that the measurement points on the surface to be inspected can be reached safely by the CMM probe with the part in its setup orientation. The inspectability criterion is determined by a two-fold test: visibility and accessibility.

offset plane

Visibility Check
2-D Line

Accessibility Check
Probe Geometry and Dimensions

Intersection
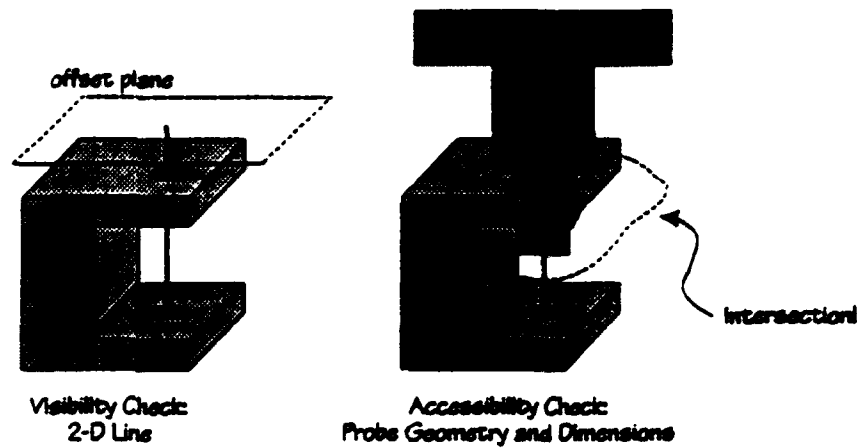
**Figure 28.** Visibility and accessibility tests place each inspection point into a setup orientation.

Visibility precedes the accessibility test since it is less computationally expensive. Visibility checks for a collision-free 2-D line originating at the inspection point, proceeding in the direction of the resting surface normal, and terminating at the offset-plane safely above the part. The point is determined to be visible if the 2-D line does not intersect the part-model. If the inspection point is visible, i.e., the visibility test returns true, then the accessibility test is executed. Actual CMM probe geometry and dimensions are used to check for possible probe head collisions (intersections) with the part-model when making the measurement. If all inspection points within an MR pass both visibility and accessibility tests, then that MR is placed in the setup.

## 4.1.3. Plan Sequencing

In a generic sense, inspection plan sequencing determines the setup orientations within the inspection plan and the DRF ordering within the setups. In a specific sense, inspection plan sequencing reduces the set of redundant setup-DRF-MR struc-

tures into a concise, unique inspection plan. The goal is to minimize the number of setups, since each new orientation requires inspector interaction in part relocation and possible fixturing.

The plan sequencer consists of two sub-processes: ordering setups, and ordering MRs within a setup. This two-step sequencing is analogous to a global ordering and a local ordering, respectively. In global ordering, the goal is to find the minimum number of setups for complete inspection of the workpiece. The algorithm searches for and begins with the setup object containing the largest number of MRs in its setup-DRF-MR structure. All identical MRs from the other setup objects are then eliminated from their duplicate setup objects. This process will result in some setups losing all their MRs, and thus that entire empty setup-DRF-MR structure is eliminated from the original set. The setup with the next largest number of total MRs is chosen and the process is repeated until no redundant MRs exist in the setup-DRF-MR structures. At the conclusion of this process, the inspection plan contains the minimum number of setups with no redundant measurements.

Local ordering involves proper sequencing of DRFs within each setup based on tolerance relationships to datums. The presence of related datums in the definition of the DRF establishes the sequencing criteria. For example, a PAP (plane, axis, plane) DRF type will often have a position callout of the secondary axis datum which is related to a PPP DRF also in that setup. Therefore, according inspection technique logic, the PAP and its related axis datum must be measured after the PPP DRF upon which it is dependent.

The result and output of the inspection plan sequencer is an inspection plan representation consisting of a minimized number of setups with stable resting surfaces, proper sequencing of dependent tolerances, and eliminated redundant tolerance

measurements. The next step is to translate this inspection plan representation into commands for the CMM probe travel and measurement evaluation to determine if the workpiece has been manufactured within tolerance.


## 4.2. Intelligent Scheduling Optimization

The previous section discussed how the IPEM creates the inspection plan by first creating the set of all allowable setup-DRF-MR structures using two measurement point criteria: visibility and accessibility. Heuristic algorithms then remove the redundant MRs and sequence the remaining MRs within the DRF objects according to their datum dependencies. Notice that the criteria for ordering the MRs within the DRFs and the inspection points within the MRs is a singular criterion based upon orientation only; relative position is not considered. In addition, the measurement points are positioned on the tolerance surface randomly, with constraints testing the validity of the placement (Table 3 on page 47). Therefore, the order of points within the current inspection plan representation has absolutely no meaning! The next task of the IPEM optimizes the path trajectory of the CMM probe by ordering the offset points throughout the DRF and MR objects. However, it is not enough to lump all the points within a setup together and optimize based solely on distance. The ordering of the offset points within the inspection plan needs to follow three inspection criteria:

> Criterion 1: GD&T rules of tolerance evaluation must be met.

> Criterion 2: the plan should be executed in a safe and efficient manner.

Criterion 3: the plan needs to be understandable to the inspector, especially since it will be automatically generated and the inspector must then interpret it.

This style of optimization problem resembles the *Traveling Salesman Problem* (TSP) which is one of the most widely studied combinatorial optimization problems [Laporte, 1992]. Several other permutation problems can also be described as TSP when the distance criterion is changed to a cost function to be minimized: computer wiring, wallpaper cutting, hole punching, job sequencing, dart board design, crystallography, etc. These research thrusts have created several solution approaches based on the desired result. Solution methods range from sequential to parallel, and exact to approximate. Sequential methods are implemented heuristic searches which look throughout the solution space, one permutation at a time. Parallel methods utilize neural networks, which have caused the area of neural networks to recently undergo a resurgence in research activity.

· This section utilizes the computation methods of the TSP to provide solutions that meet the criteria of inspection planning. Both the heuristic search and neural network approaches are implemented within the IPEM. The heuristic search method implements the nearest neighbor algorithm with a two-level hierarchical search modification. The artificial neural network implementation uses a Hopfield network for optimization with an inspection rule generated cost function modification. However. each method requires a different inspection plan structure which significantly effects the operations of the automated code generator.

## Plan Structures Dictated by Point Order

All automated inspection plans require a specific format to properly inspect the workpiece. The CMM instructions must contain the following modules in order:

1. instruct the inspector to place the workpiece in the initial/next setup to inspect [INIT]

2. instruct the inspector to locate the workpiece by manual inspection, creating the local coordinate frame [SETUPa]

3. (optional) re-measure the local coordinate frame automatically for better accuracy [SETUPb]

4. if different than the initial coordinate frame, measure the datums and mathematically create the next DRF [DRF]

5. measure a tolerance within that DRF [MR]

6. show evaluations when the measurement is completed [EVAL]

7. return to 1 if there exists a next setup; otherwise, end [DONE?]

However, there are three flow possibilities that provide various degrees of efficiency versus readability in the resultant automated inspection plan. Each flow possibility is the result of the criteria used to sequence the inspection points.
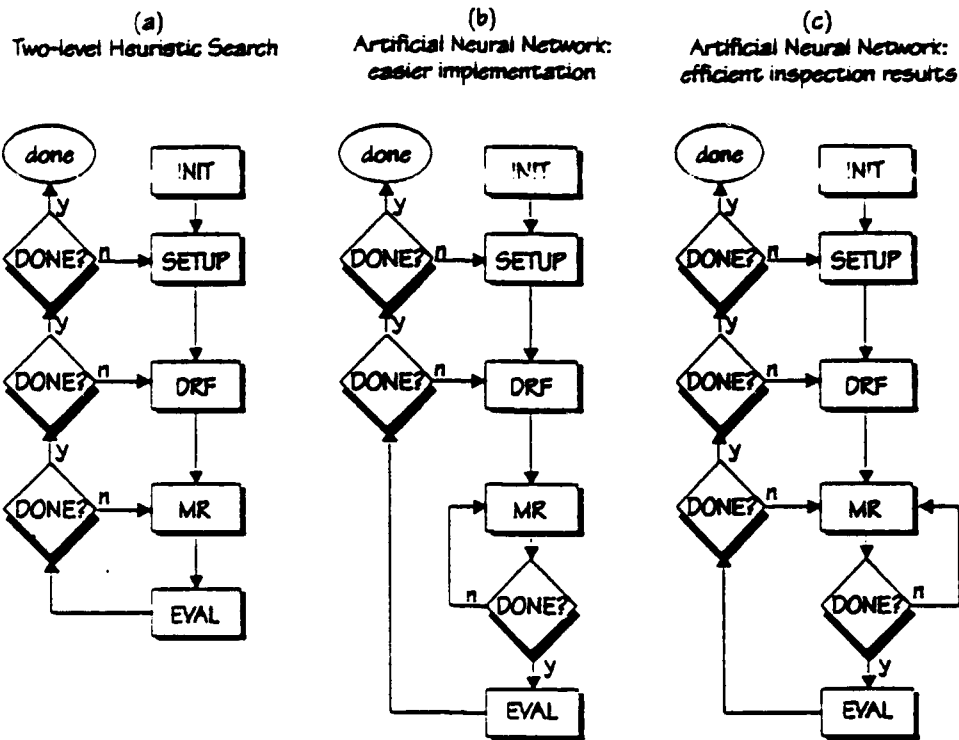
**Figure 29.** Three CMM program code structures.

Figure 29(a) shows CMM instruction command structure if the two-level nearest-neighbor sequence algorithm is used (section 4.2.1). Its advantage is inspector readability. Each tolerance evaluation is performed directly after the measurements are taken, and all measurements of an MR are performed together. This is advantageous for small lots of different workpieces.

Figures 29(b) and (c) show the command structure of using the ANN scheduling (section 4.2.2) where the inspection points among the different MRs are intermixed within the inspection plan schedule. The evaluation command cannot occur until all its inspection points have been probed. Figure 29(b) shows the least difficult way for the IPEM to generate the code, but it is also less efficient. In this

scheme, all the inspection points are probed and saved into memory. Then all the evaluations occur, each evaluation function retrieving the proper measurements from memory. This makes it hard for the inspector to interpret the automated inspection plan. This is advantageous for large lots of the same workpiece.

Figure 29(c) shows the more efficient way to perform the command structure. An evaluation is performed after a successful check is made to see if all the inspection points needed to perform the evaluation have been previously measured. This provides immediate analysis of the tolerances on the manufactured part. One advantage is that when the tight or critical tolerances are measured first, the inspection plan can be terminated and cease performing unnecessary measurements. For example. if a datum is out of position tolerance, then all the points that reference that datum cannot be measured properly. Once the datum is evaluated, the CMM program can stop the plan and prompt the inspector to continue.

## 4.2.1. Heuristic Search Scheduling

Broadly speaking, optimization searching heuristics can be called tour construction procedures which gradually build a solution by adding a new vertex at each step based on a pre-defined criterion [Laporte, 1992]. The solution methods can be classified into exact and approximate searching. The algorithm for exact solutions searches the entire solution space. applying the criterion heuristic to all allowable solution combinations and then selecting the best result once the entire space is searched. This approach requires unreasonable time and memory to find the optimum solution. even when the problem size is small. Approximate methods do not carry the guarantee of an optimal solution. but rather seek to find a "good" or acceptable solution within minimum effort. Therefore, this research rejected the

implementation of an optimum solution heuristic search and chose the approximate method for several reasons. The computational time and resources to obtain the optimal solution were deemed unnecessary, and an approximate solution will meet the needs of the inspection plan, i.e., the inspection criterion #2 (page 52) called for an *efficient* path, not an *optimum* path. The workpieces are relatively small in size and measurements relatively close, so that the excess probe traveling time of an approximate solution versus an exact solution is inconsequential compared to other inspection tasks, such as probe recalibration, setup orientation, and manual part location. For these reasons, approximate solution methods were considered as being appropriate. These reasons show that the shortest distance is not a top priority.

Why is it stressed that an optimum path is not critical? Consider the way in which people negotiate their way across rooms cluttered with tables and chairs. It is unlikely that they reliably chose the very best route; however, they seldom chose a very bad one [Pratt, 1991]. With the destination known and knowledge that several good paths exists, the need for an optimal path is not necessary. For this same reason, sensible paths are appropriate for this research solution. Another reason is that the most important factor to automated robot motion planning is collision-free, safe paths. In fact, making a path collision-free requires extending the path around obstacles which increases path length. Once the collision-free path is created by the insertion of via-points (section 4.3.2), the originally o, ̃mum point sequence might no longer correspond to the minimal path. Section 7 discusses the desire for incorporating collision-avoidance methods into the point sequencing rules, but currently no one is known to do this.

There are many approximate heuristic search algorithms designed to solve the TSP: nearest neighbor, insertion, asymmetrical patching, r-opt, simulated annealing,

tabu, etc. [Laporte, 1992]. The IPEM implemented the nearest neighbor algorithm. The advantage is its simplicity and speed of solution, while its drawback is its myopic view of the problem. The path is constructed by, at each step, taking the decision that is immediately the most advantageous, defined by the minimum distance among the current point and all the other remaining points. Once a point is placed in the path, the point is removed from consideration as a remailing available point. Appropriately, another name for the nearest neighbor algorithm is the greedy algorithm. The general structure for the algorithm is:

> Nearest Neighbor:
>     Consider an arbitrary vertex as starting point
>     Repeat
>         Determine the closest vertex to the last vertex considered and
>             include it in the tour
>     Until no vertex has not yet been considered
>     Link the last vertex to the first one

The time complexity of this algorithm is $O(n^2)$.

## Modifications to Nearest Neighbor Search

The generic nearest neighbor algorithm is modified to make it meet the three inspection plan criteria indicated above. The first criterion requires the GD&T rules of tolerance evaluation to be met. One GD&T rule is that the related tolerances must be measured with respect to the datums of the datum reference frame. Therefore, each setup-DRF structure is considered its own individual sequencing sub-problem. This GD&T rule also implies that the inspection points of the datums that make the DRF are removed from scheduling consideration since they must be measured first in order to create the DRF before any tolerances can be measured. A final implication

is that the initial point of the nearest neighbor algorithm should not be chosen randomly, but rather should be selected from the tertiary datum.

It should be noted here that a CMM evaluation rule also prohibits the sequencing of the inspection points of the datums using the 3-2-1 DRF creation. The measurements of the primary and secondary inspection points must occur in a specific order, regardless of distance. The direction of the normal of the primary, secondary, and tertiary datums are calculated by applying the right-hand rule to the three points, starting with the first, then second, then third measurements or calculations. This approach is described in greater detail in section 4.3.1.

The second inspection criterion (page 52) is partly met by this algorithm. An efficient path is sought by means of the general nearest-neighbor search; however, a safe path is not considered. The collision avoidance algorithm is discussed in section 4.3.2.

. The third inspection criterion might be the most overlooked aspect of automated process planning systems — inspector ease of use. Since the RDS is designed to help humans perform their job better, an output that requires extensive interpretation by the inspector for it to be understood will defeat its own purpose and worthiness. The two-level hierarchical search modification to the nearest-neighbor algorithm implements the thinking process of the inspector into the search. Since the inspector visualizes the inspection plan in terms of evaluation results of a set of measurements, all required inspection points within a tolerance evaluation are measured together at once, and the evaluation is performed immediately after the measurement of the last point. The alternative method measures the inspection points in accordance with a minimum distance criterion. regardless of which tolerance evaluation method the measurement point was created for. Once measured, the

coordinate values are saved into memory, and all the evaluations are performed at the end of the measurement sequence. This method results in a confusing and unfriendly plan representation for the inspector.

Using the two-level hierarchical search provides benefits for both the second and third inspection criteria. First, the inspector can see before and during the inspection why a point is being measured: each evaluation occurs directly after all inspection points are measured, and all inspection points needed for the evaluation function are measured sequentially, and not intermixed. The learning curve of interpreting the automatically generated inspection plan is significantly reduced. Second, the evaluation functions by the CMM language operate immediately on the completion of measurements and not at the end of the plan. This allows the ability to take advantage of go/no-go evaluation techniques. For example, by placing the tightly toleranced or *a priori* difficult-to-manufacture features to be completely measured first, the inspection process can promptly quit when an evaluation fails to be within tolerance, thus eliminating unnecessary probing. A future consideration for the IPEM is to utilize statistical process control (SPC) information to learn which tolerances are "important" based on previously inspected workpieces.

## Implementation

The heuristic search begins at the tertiary datum of the DRF. Since the 3-2-1 DRF creation method is used, the tertiary datum is just a single point. The top-level search executes, choosing the nearest neighbor to the tertiary datum from among all the inspection points in the DRF, no matter with which MR it is associated. The MR that contains this nearest-neighbor point is sequenced as the first MR in the DRF. The algorithm now begins the second level of the hierarchical search. A nested

greedy search starts anew using the nearest-neighbor point from the top-level search as the initial point for the lower-level search. The only valid inspection points for this search are the ones contained within the current MR, i.e., the MR chosen from the first level. Once the nearest-neighbor ordered path is completed for the MR's inspection points, the lower-level search returns the last point of the ordered point set to the top-level search, where it replaces the current point selected before the lower-level search. The top-level search continues the nearest-neighbor algorithm with the replaced point from the end current MR. Obviously, once an MR and its inspection points are optimized in the lower-level, they are no longer eligible within the algorithm for nearest-neighbor consideration any longer.
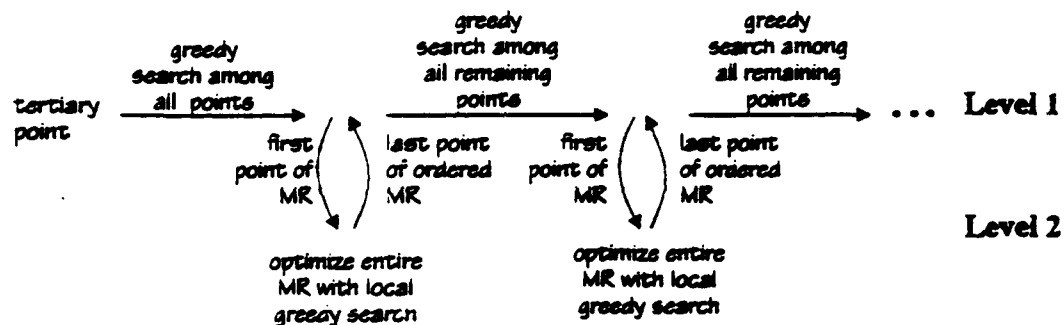
Figure 30. Flow diagram of the two-level nearest-neighbor algorithm.

## Results

The results of the two-level hierarchical nearest-neighbor algorithm are shown to be acceptable. Figure 31 is a snapshot from the IPEM display window with a part-model that is designed after an actual part received by the QA engineers at the 4950th Test Wing. The tertiary point denoted with a "circled $t$" beside it is the starting

point for the algorithm. The path is seen as sensible, and is actually the global minimum.
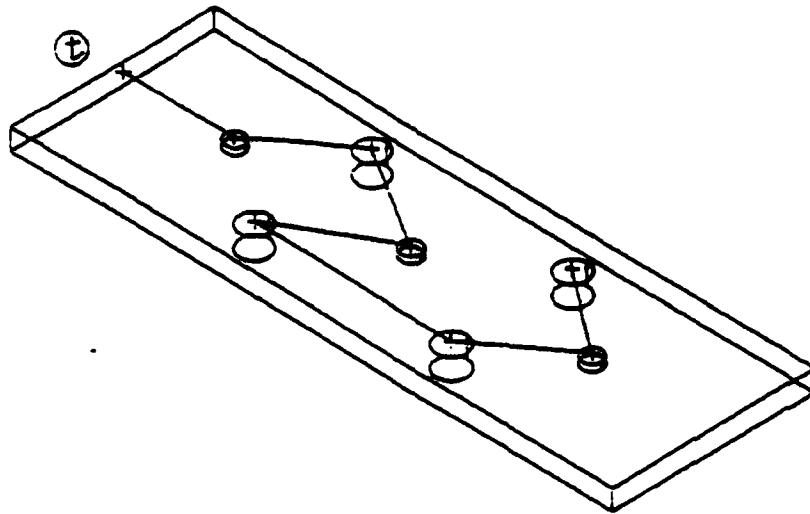


**Figure 31.** Illustration of two-level nearest-neighbor search.

. The path generated using this method is not always the global minimum. In fact, more often than not, that is not the case. The part-model in Figure 31 contained only MRs with one point to be considered for sequencing. As the MRs contain more inspection points, the sequenced path will deviate from the minimum path. Figure 32 shows an example result of this situation. Notice that the path does cross over itself several times, but there is an ordered flow to it across the product that the inspector will recognize and appreciate: the probe travels completely throughout an MR before going to the next closest path.

second MR chosen
(5 pts)

third MR chosen
(3 pts)

first MR chosen:
closest to tertiary
datum (6 pts)

tertiary datum:
starting point

last MR chosen
(1 pt)

fourth MR chosen
(1 pt)
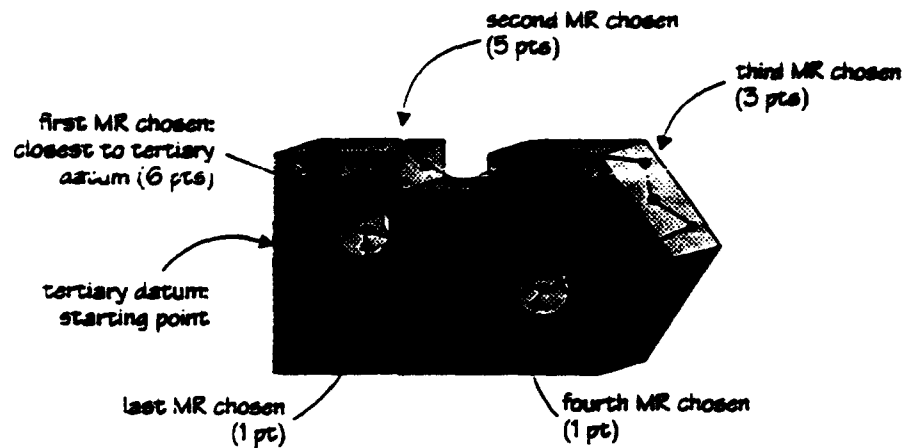
**Figure 32.** Illustration of two-level nearest-neighbor algorithm with multiple inspection points per MR.

## 4.2.2. Hopfield Net Scheduling with a Rule-Based Lyaponov Function

In 1985, J. J. Hopfield and D. Tank showed that artificial neural networks (ANNs) could be used to solve complex combinatorial optimization problems by solving the 10-city TSP problem. The results were actually slower than a popular heuristic search [Xu and Tsai, 1991]. However, the use of artificial neural networks has become very attractive due to inexpensive VLSI (very-large-scale-integrated) circuit technology [Takefuji, 1992]. Many researchers have improved upon the original Hopfield energy function to obtain worthwhile results. Abe, et al., (1992) says that the Hopfield neural network is "well suited to obtain near optimal solutions for combinatorial optimization problems."

Takefuji's book discusses the implementation of neural networks for problem solving and has a stated intention to "demonstrate the capability of the artificial neu-

ral network for solving optimization problems over the best known algorithms or the best methods if they exist." One of the reasons for such successful results is the modeling of the energy functions and motion equations that are used to drive the network to the solution.

The approach in this research, to implement a neural network for inspection point placement, is adapted from the Hopfield neural network model. However, the goal is the intelligent ordering of inspection points based upon the three basic inspection criteria (section 4.2), not just a minimum distance. The approach to meet the inspection criteria is accomplished through rule-based weight matrices implemented within the Lyaponov energy function of the neural network. The weight matrices, created within the IPEM, are based upon desired inspection techniques and rules that the inspector would use for inspection point sequencing throughout the inspection plan. Using inspection rules within the motion equation of the neural network will produce an output sequence that is optimum for the task of inspection planning.

## Neural Network Representation for Optimization

The implementation of neural networks involves modeling the neurons to represent the finite elements of the problem and giving the neurons a motion equation to guide them to the desired solution. In optimization scheduling, a neuron represents a task $i$ being performed at time $j$. Each neuron has input $U$ and output $V$. For a problem with $n$ tasks with each task to be performed once, the set of all possible states is represented by $V_{ij}$ for $1 \leq i \leq n$, $1 \leq j \leq n$, in an $n \times n$ 2-D array.

j (order number)



solution = 3,1,6,4,7,2,8,5
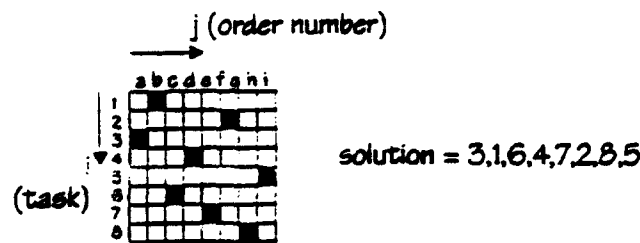
(task)

**Figure 33.** Representation of neural net result.

A blackened cell represents a neuron that is "on" or $V_{ij} = 1$ and task $i$ is set to be performed at time $j$. An empty cell represents a neuron that is "off" or $V_{ij} = 0$. The goal is to find a state of the system that represents a valid solution of the problem. A solution must satisfy all of the following constraints:

1. all tasks must be performed only once: $\sum_{k} V_{ik} = 1$

2. each time slot can perform only one task: $\sum_{k} V_{kj} = 1$

3. all tasks must be performed: $\sum_{i} \sum_{j} V_{ij} = n$

4. the solution must minimize a given cost function

ANNs are a mathematical representation of the neurological functioning of the human brain. The network is structured as a massively parallel array of simple interconnected processing elements called neurons, representing the input/output function of the biological neuron of the brain. The function of the neuron is to take the inputs of several independent signals and produce an output. The input signals are generated from the outputs of the other neurons and propagated by synaptic links. The neurons are given an input/output function $V = f(U)$ that models the function

biological neurons are known to possess. Figure 34 displays different implementations and approximations to the neuron activation function.



$$V = f\left(\sum_{k}^{n} w_k \cdot V_k\right)$$

**Sigmoid**

$$f(U) = \frac{1}{1 + e^{-(U - \theta)/\theta_0}}$$

**McCulloch-Pitts**

$$f(U) = \begin{cases} 1 \text{ if } U > 0 \\ 0 \text{ if } U < 0 \\ \text{unchanged otherwise} \end{cases}$$

**Hysteresis McCulloch-Pitts**

$$f(U) = \begin{cases} 1 \text{ if } U > \text{UTP (upper trip point)} \\ 0 \text{ if } U < \text{LTP (lower trip point)} \\ \text{unchanged otherwise} \end{cases}$$
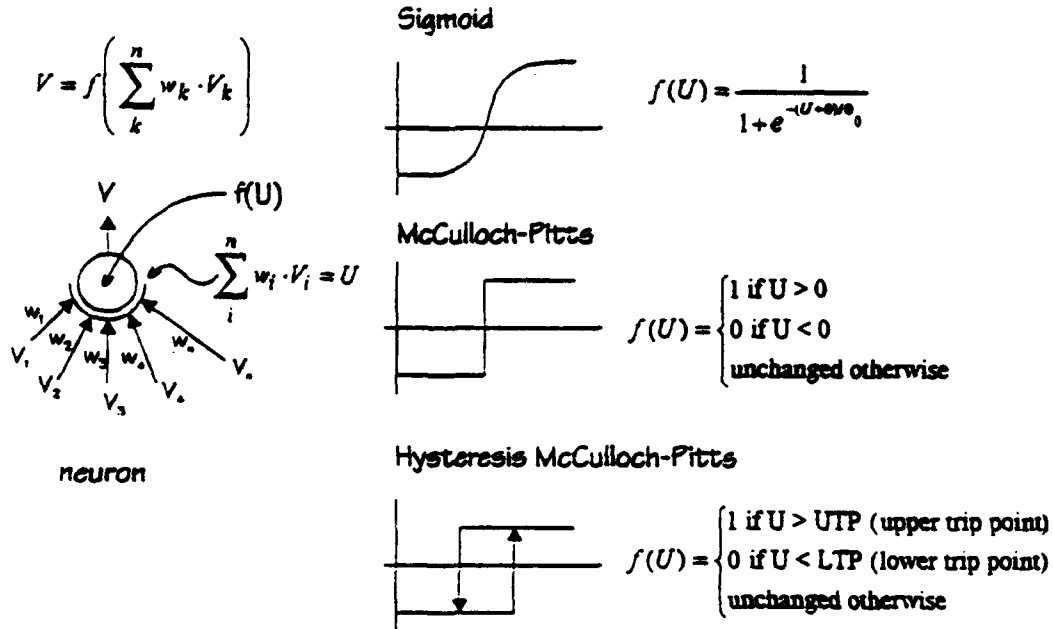
**Figure 34.** Neuron representation and input/output functions.

The goal of the ANN is to solve the optimization problem by providing a parallel gradient descent method to minimize the system energy. The system energy is a function of the neuron outputs, $E(V_1, V_2, \dots V_n)$, defined by the programmer to best solve the optimization problem. The change in the value of the input state of the $i^{th}$ neuron is determined by the partial derivatives of the energy function with respect to its output.

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, V_2, \dots, V_n)}{\partial V_i} \tag{4.5}$$

This is called the motion equation of the $i^{th}$ neuron, since it defines how the energy will propagate through the network — towards minimizing the computational energy.

Takefuji (1992) states that it is usually easier to build the motion equation and then derive the energy equation from the integration of equation (4.5):

$$E = \int_j dE = -\int_j \frac{dU_i}{dt} dV;$$ (4.6)

In their original paper, Hopfield and Tank (1985) defined the neuron motion equation as:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i}$$

$$= \frac{-u_{xi}}{\tau} - A \sum_{j \neq i} V_{xj} - B \sum_{X \neq Y} V_{Yi} - C \cdot \left( \sum_X \sum_j V_{Xj} - n \right)$$ (4.7)

$$- D \sum_Y d_{XY} \cdot \left( V_{Y,j+1} + V_{Y,j-1} \right)$$

The use of the decay term $(-U_i/\tau)$ is now known to hurt the dynamics of the network [Takefuji, 1992]. Removing the decay term produced more efficient motion equations and enhanced the performance of the ANN to solve optimization problems. Equation (4.8) shows a motion equation which is a slight update from the original Hopfield model in equation (4.7), and is generally accepted as the standard motion equation for Hopfield network implementation for solving the TSP optimization problem [Xu, 1991][Jeon, 1990][Abe, et al., 1992].

$$\frac{dU_{ij}}{dt} = -A \left( \sum_k^N V_{ik} + \sum_k^N V_{kj} - 2 \right) - C \cdot h \left( \sum_k^N V_{ik} \right) - C \cdot h \left( \sum_k^N V_{kj} \right)$$

$$- D \left( \left( \sum_k^N V_{k,j+1} + V_{k,j-1} \right) \cdot W_{ik} \right)$$ (4.8)

The $A$ summation term monitors the first and second constraints stating that each row and column should have only one neuron fired, i.e., "on". The two $C$ summation terms are Takefuji's (1992) hill climbing term implemented to help the

network motion to escape local minima. The final $D$ summation term provides the criterion to be minimized. $W_{ik}$ represents the Euclidean distance in the TSP problem, or can be defined to represent any cost function modeling the problem to be minimized.

The neuron energies are then updated by the first order Euler method:

$$U_{ij}(t+1) = U_{ij}(t) + \frac{dU_{ij}(t)}{dt} \cdot \Delta t \qquad (4.9)$$

where $\Delta t = 1$, representing one time step. The process of running the network is a simple loop terminated by a system state check:

<u>ANN:</u>
    Initialize each neuron input with a random number
    Repeat
        Evaluate the output of the neuron based on the neuron
            input/output function $V = f(U)$
        Update the inputs to each neuron from the motion equation,
            equation (4.8) and equation (4.9)
    Until there is no more than one neuron on in any row or column
        or the number of iterations is greater than the maximum allowed

## Lyaponov Functions

The purpose of a Lyaponov function is to provide a shortcut to proving global stability of a dynamic system [Kosko, 1992]. In general, the Lyaponov approach reveals only the existence of stable points, not their number or nature.

A Lyaponov function $L$ maps system state variables into real numbers. A system is called stable if $L$ decreases along its trajectories with respect to time. This is proved by showing that the change in $L$ with respect to the change in time is negative for all time:

$(\Delta L/\Delta t) < 0$; for stable system $L$,

$(\Delta L/\Delta t) \leq 0$; for asymptotically stable system $L$. (4.10)

Used mostly in control systems, the Lyaponov function has been applied to neural network analysis to provide insight into the stability of the network's energy state. Consider the time derivatives of the neural network energy function:

$$\frac{dE}{dt} = \dot{E} = \sum_i \frac{dV_i}{dt} \frac{\partial E}{\partial V_i}$$

$$= -\sum_i \frac{dV_i}{dt} \frac{dU_i}{dt} \qquad (4.11)$$

$$= -\sum_i \left( \frac{dV_i}{dU_i} \frac{dU_i}{dt} \right) \frac{dU_i}{dt} = -\sum_i \left( \frac{dU_i}{dt} \right)^2 \frac{dV_i}{dU_i}$$

The first line uses the chain rule to get the time derivative of $E(V_1, V_2, \dots V_n)$. The second line is obtained from the substitution of equation (4.5) into the first line. The third line is a chain rule expansion of the $(dV_i/dt)$ term and then simplification. Since $(dU_i/dt)^2$ will always be positive (due to the squaring), the negation of the sum of squares will always be negative. Therefore the partial derivative of the neuron input/output function $(dV_i/dU_i)$ must be negative for equation (4.10) to be valid. The McCulloch-Pitts function (Figure 34 on page 66), chosen as the neuron input/output function for this research, has its change in output with respect to the change in input that is always less than zero (a negative slope). Therefore, the energy derivative $(dE/dt)$ is less than zero and the neural net is stable.

## Implementation

Two main accomplishments provided the implementation of the rule-based Lyaponov function within a Hopfield neural network for optimization. The first ac-

complishment created a set of rules that portrayed the desires of the inspector in the inspection plan creation. The rules are written in the IPEM and are based on relationships held between the inspection points and their MRs and surfaces. Three different types of matrices were created from the inspection rules to be used by the ANN.

The second accomplishment implemented an ANN that would optimize the measurement point sequence based on the inspection rule matrices created within the IPEM. The software implementation of the ANN began with the work done by Jeon (1990), which was designed for the purpose of inspection planning techniques, but only addressed the issue of minimum distance and not inspection criteria. His setup is identical to the two-level nearest-neighbor heuristic search discussed in section 4.2.1, except that neural network optimization replaced the lower-level greedy search. His network optimization was limited to singular surfaces as well as singular features. Therefore, several modifications to Jeon's neural network were made to allow the implementation of the inspection rules. The neuron motion equation and update functions were changed to accept the three inspection-rule-created matrices from the IPEM.

## Implementation: Creating the rules in LISP

Two different forms of rules were realized to represent a preference in inspection point sequencing. The first form relates a point to its overall sequence position in the inspection plan. For example, one inspection rule states that all "tight" tolerances must occur at the beginning of the sequence of measurements. The definition of "tight" is defined by the inspector, but the IPEM currently uses a value of 0.005 inches. An $S$ matrix (mnemonic for Sequence) represents this form of rule by telling the neural net whether a point $i$ at sequence $j$ is allowed to exist according

the inspection rules. $S_{ij} = 1$ if the sequence position is allowed, and $S_{ij} = 0$ if it is

forbidden. This $S$-rule is implemented by counting the total number of tight

tolerances and setting $S_{ij} = 1$ for each tight tolerance $i$ and $1 \leq j <$ (total number of

tight tolerances); otherwise, $S_{ij} = 0$. For each non-tight tolerance $i$, $S_{ij} = 1$ for (total

number of tight tolerances) $\leq j < n$; otherwise, $S_{ij} = 0$.

One other $S$-rule also exists. It states that the tertiary tolerance must be the first

sequence point, and that no other point is allowed to be first. This requirement is

easily implemented by setting $S_{00} = 1$ and $S_{0i} = S_{i0} = 0$ for $1 \leq i < n$. Notice that the

matrix indicators are zero-based, to represent their implementation in both LISP and

C software languages.

S matrix created by two S-rules: tertiary and tight

total number of tight tolerances (3)

| | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| tertiary | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| pt1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| pt2 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| pt3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| pt4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| pt5 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| pt6 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| pt7 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

tight → pt2, tight → pt5, tight → pt6

**Figure 35.** Populating the $S$ matrix.

The second form of inspection rule relates one inspection point to another.

Two matrices fulfill this type of rule. The $W$ matrix (mnemonic for Weight) is the

standard distance matrix seen in TSP problems. $W_{ij}$ represents the Euclidean dis-

tance from point $i$ to point $j$. The $F$ matrix (mnemonic for Follow) represents the

rule-based penalty value for point $i$ being followed by point $j$ at *any* two sequential time slots in the inspection plan. For example, one $F$-rule states that if two points, $i$ and $j$, are from the same MR, then $F_{ij}$ should be some negative value (good), otherwise $F_{ij}$ should be some positive value (bad). The inspection rules currently implemented create a cumulative penalty value for $F_{ij}$ based on both good and bad reinforcements:

bad (+)   $i$ and $j$ are from different MRs
bad (+)   $i$ and $j$ are on different surfaces
bad (+)   $i$ and $j$ are different types of features
bad (+)   $i$ and $j$ are different types of tolerances
good (-)   $i$ and $j$ are the same feature type
good (-)   $i$ and $j$ are the same feature type and have the same
              feature dimensions
good (-)   $i$ has a tighter tolerance than $j$
bad (÷)   $j$ has a tight or critical tolerance and $i$ does not
good (-)   both $i$ and $j$ have tight tolerances

The inspector can vary the relative strength of the individual inspection rules to customize the rules to produce the desired output. The rule strength parameters allow some rules to be weighted more heavily than others by changing individual weights that pre-multiply each rule before it is summed into $F_{ij}$. A future user interface will give the inspector easy access to the rules allowing him to manipulate the outcome of the schedule solution as desired.

Notice that some inspection rules can be implemented in both the $S$ and $F$ matrices. This is advantageous to the ANN. By telling the ANN similar rule information in different formats, the search criteria becomes more robust resulting in a quicker convergence.
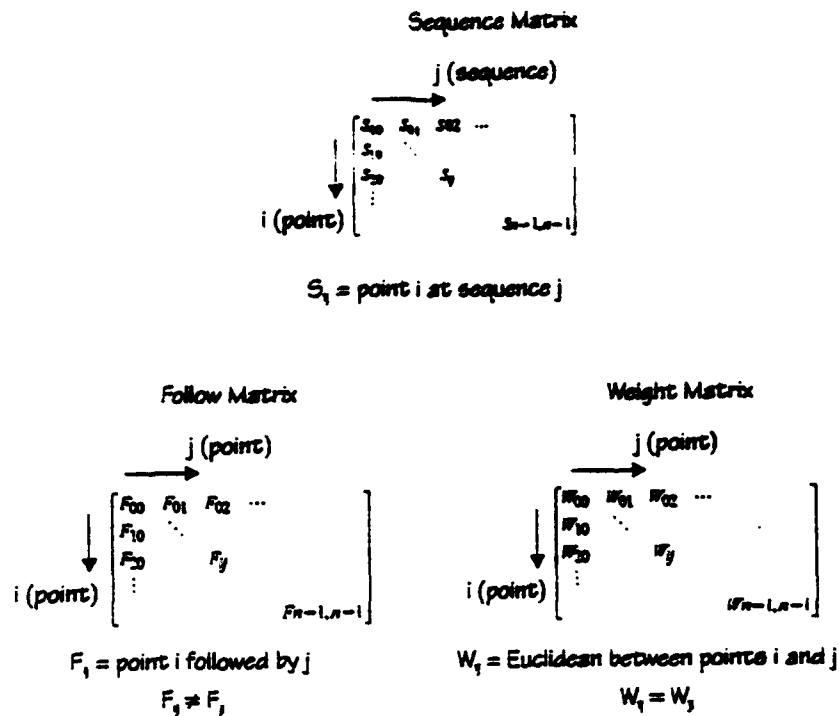
**Figure 36.** The *S*, *W*, and *F* matrices used to implement inspection rules into the neural network.

Below is an example of a rule implementation as executed from within the IPEM.

```
;; tight tolerance -- f represents F[i][j]
(cond ((and (<= (tol-value@ pti) tight-tol)
            (<= (tol-value@ ptj) tight-tol)) ;i&j tight
       (setf f (+ f (* Wtight good))))
      ((and (>= (tol-value@ pti) tight-tol)
            (<= (tol-value@ ptj) tight-tol)) ;j tight, i not
       (setf f (+ f (* Wtight bad))))
      ((and (<= (tol-value@ pti) tight-tol)
            (>= (tol-value@ ptj) tight-tol)) ;i tight, j not
       (setf f (+ f (* Wtight good))))
      (t (setf f (+ f (* Wtight ok)))))) ;otherwise
```

Notice that the rules contain both good and bad reinforcements. Since the $\Sigma F_{ij}$ will

be utilized in the fashion of a distance penalty which is minimized by the ANN,

good $\cong$ -1, bad $\cong$ 1, and ok $\cong$ 0.

Once the matrices have been created, they are post-processed for use by the ANN. Each matrix is normalized so that every value in the array is between zero and one. The $W$ matrix also has minimum shift performed on each of its rows. For each measurement point (row) in $W$, the smallest distance to each of the other points (columns) is determined and subtracted from each of the points in that row. This leaves the relationship between the points constant, but the minimum distance is now 0.00. This speeds up the convergence of the ANN.

## Implementation: Utilizing inspection rules in the ANN

Several modifications to the ANN model described above were performed to incorporate inspection rules into the solution selection. The first modification requires that the tertiary tolerance always be sequenced first and at no other sequence position, and that no other tolerance occupy the first sequence position. These requirements are accomplished by setting $v_{00}=1$ and $v_{ij} = 0$ for $i = 0, j \neq 0$ and $j = 0$, $i \neq 0$ within the neuron input/output evaluate function. This approach is implemented in addition to the $S$-rule mentioned above to provide faster convergence.

In order to shorten the convergence time, the energy input values are kept within fixed limits. Therefore, when $U_{ij}(t+1)$ is calculated and $t$ updated, the value of $U_{ij}(t)$ is evaluated by:

$$U_{ij} = \begin{cases} HL \text{ if } U_{ij} > HL \text{ (high limit)} \\ LL \text{ if } U_{ij} < LL \text{ (low limit)} \\ \text{unchanged otherwise} \end{cases} \qquad (4.12)$$

This approach provided a significant improvement in the run-time of the ANN. When these thresholds were used, the number of iterations required to reach a solution decreased by two orders of magnitude!

The actual neuron motion equation (4.13) was also modified to accommodate the new inspection rule information. The $W$ matrix remains the distance penalty from before. The $F$ and $S$ matrices are also treated in the same manner as distance penalty functions. The $S_{ij}$ value is added to the update equation rather than subtracted, since it was defined as a positive value for preferable and zero for not preferable.

$$
\begin{aligned}
\frac{dU_{ij}}{dt} = &-A \cdot \left( \sum_{k}^{N} V_{ik} + \sum_{k}^{N} V_{kj} - 2 \right) - B \cdot \left( \left( \sum_{k}^{N} V_{k,j+1} + V_{k,j-1} \right) \cdot W_{ik} \right) \\
&-C \cdot \sum_{\substack{p=1 \\ p \neq q}}^{n} \sum_{q=1}^{n} V_{p,j-1} \cdot V_{p,j+1} \cdot \left( W'_{ip} + W'_{iq} + W_{pq} \right) \\
&+ D \cdot h \left( \sum_{k}^{N} V_{ik} \right) \cdot h \left( \sum_{k}^{N} V_{kj} \right) \\
&- F1 \cdot \sum_{\substack{k=1 \\ k \neq i}}^{n} \left( V_{p,j-1} \cdot F_{pi} + V_{p,j+1} \cdot F_{ip} \right) \\
&+ S_{ij}
\end{aligned}
\tag{4.13}
$$

The $A$ summation term monitors the same row and column constraints as discussed in equation (4.8) on page 67. The $B$ summation term is the standard Euclidean distance term from Hopfield (1985) and is equivalent to the $D$ summation term of equation (4.8). The $W'$ is the distance matrix prior to the minimum shift operated on each row. The $C$ summation term prevents the system from taking roundabout paths, and is discussed in greater detail by Jeon's thesis (1990). The $D$ summation term represents the hill-climbing function that helps the system escape local minima. It is equivalent to the $C$ summation term of equation (4.8). The $F1$ summation term represents the penalties from the $F$-rules according to the current system status of $V_{ij}$ with respect to the points scheduled before and after it. As mentioned above, the $S_{ij}$ term is simply

added into the neuron update equation to represent the inspection rule of point $i$ occurring at sequence order $j$.

One inspection rule was implemented within the ANN to update the $F1$ coefficient according to the state of the system. The $F_{ij}$'s are created based on the $F$-rules which state that a point with a certain characteristic, $c_1$, should not follow a point with a different characteristic, $c_2$. However, at some moment in the inspection plan, the sequence of inspection points will have sequenced all the points with characteristic $c_1$ and will then switch to points with characteristic $c_2$. This switch over is preferable and should not be penalized. Therefore, the $F1$ coefficient is updated by an inspection rule within the ANN which sets $F1$ equal to zero if the $F$-rule penalty is not desired because all other $F$-rules are satisfied.

## Results

A comparison is made between the two sequencing optimization techniques discussed in this section. Figure 31 on page 62 shows the result of a two-level nearest-neighbor search algorithm on a part-model created within the RDS and processed within the IPEM. The same part is shown in Figure 37 to illustrate the greater power of the rule-based ANN. The four larger through-holes near the edges of the part-model have tight position tolerances with a tolerance value of 0.0005 inches. Using the $S$ and $F$ matrices, the ANN places these four holes at the beginning of the inspection plan, then sequences the remaining tolerances. While meeting the inspection rules, the ANN also provides the minimum distance path.
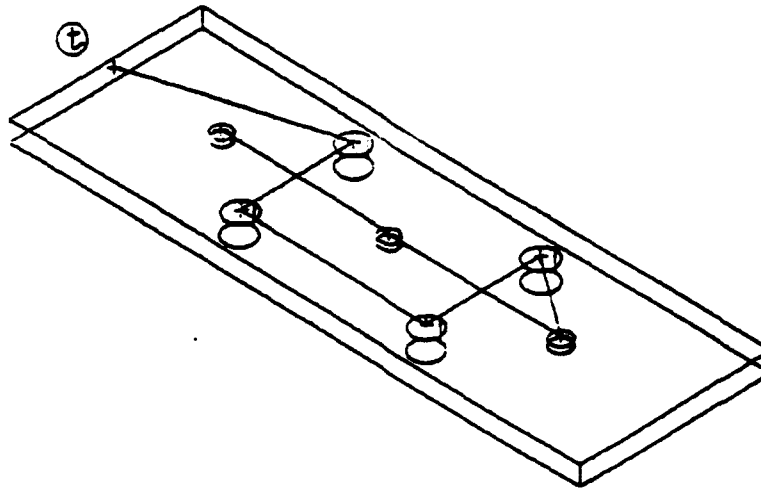
**Figure 37.** Illustration of rule-based ANN sequence result.

The ANN reached the solution in 177 iterations, which is almost no time on a SPARC2 workstation! The speed, accuracy, and thoroughness of the rule-based ANN make it a superior solution method for scheduling optimization. The output path represents a sequence readable to the inspector, since they are created by the same rules that the inspector would use.

## 4.3. Automated CMM Code Generation

The IPEM is intended to be a complete automated inspection system. Every aspect of the inspection plan creation and execution is intended to be aided to enable the engineer to do the same quality job faster. One particularly tedious job for any QA engineer is programming the CMM for automated inspection. Due to the rather recent introduction of computer-controlled CMMs, there has not been a standard language until very recently. As a result, each CMM manufacturing company has

been designing its own software language to interface its hardware interpreter. Since the strengths of this type company are usually in the hardware rather than software, the language is usually low-level and cryptic. Two unpleasant situations result. First, the commands are hard to remember because they are usually mnemonic, and they are especially hard to use since such low-level languages have little or no branching, looping, or variable capabilities. Second, since the language is controlling an expensive sensing device, an error is extremely critical. One typographical error, added digit, or overlooked minus sign can send the probe crashing into the work-piece, table top, fixturing, or itself. The cryptic commands, low-level programming, and point values that must be accurate to thousandths of an inch exacerbate the potential human error factor.

The automated code generator produces the desired output of the IPEM — executable CMM instructions to an ASCII text file that, when downloaded to the CMM controller, will perform the automated, efficient, and safe (no collisions) inspection plan to determine if the part is within tolerance. The processes involved in automated code generation are path planning for an efficient and collision-free probe path (4.3.2), feature translation from the process planner into a metacode representation (section 4.3.4), and code generation from the metacode features (section 4.3.5).

## 4.3.1. The CMES Language

The CMM used in QA procedures of the 4950th Test Wing is made by LK Tool USA. Inc. and uses a proprietary language called CMES (Co-ordinate Measuring Software). The CMES manual describes the language in not so user-friendly terms:

CMES responds to mnemonic commands known as *command codes*. [T]hese codes comprise two characters which usually describe the function of the command. [F]or example, ID and PT represent the Inside Diameter and Point commands, respectively. In order to increase the capability of each command, most CMES commands are equipped with *command parameters* which are used to cause the command to function in a specific manner.

An example of a block of CMES code to inspect just one hole and one slot, with the datums already created:

```
#MC,X,Y\10.5\-52.46
#MC,Z\-2
#PP,Y\-58.5\SP,10\#PP,Y\-48.75\SP,11
#MC,Z\10
UP,10,11\LI,Y//\10.25,.005,-.002
#MC,X,Y\12.5\-35.00
#MC,Z\-1.5
#ID,ZM\3.45,.001,-.015\.200
#MC,Z\10.00
```

Appendix B contains a quick reference to CMES code. It is placed there for three reasons. First, there are examples of CMES code throughout this paper that the reader may want to try and decipher. Second, it will help the novice reader understand the work of the inspector and appreciate the value of an automated code generator. Third, the reader who knows other software languages will be able to see the low-level of CMES, and will allow the reader who knows other inspection languages to compare between them.

As mentioned in section 4.2.1, there are also CMES guidelines incorporated into the rule-based ANN. The creation of the datums is done by the CMES commands AX, N1, N2, PI, and MD. These commands use the order in which the measurements are taken to determine the direction of the normal of the surface created with these commands. Starting at the first measured point and rotating through to the third measured point, AX uses the right-hand rule to define the

positive direction of the normal of the plane defined by the three points. The N1 command defines the positive direction of the secondary axis using the right-hand rule on the two measured points and a third point calculated along the normal of the primary axis (from AX) using the two measured points as the base. The N2 axis positive direction is constrained by:

$$\hat{x} \otimes \hat{y} = \hat{z}$$
$$\hat{y} \otimes \hat{z} = \hat{x} \qquad\qquad (4.14)$$
$$\hat{z} \otimes \hat{x} = \hat{y}$$

Since the inspection point measurements are created with a pre-defined coordinate axis system, the order of the datum plane measurement points must not change during inspection point scheduling optimization.
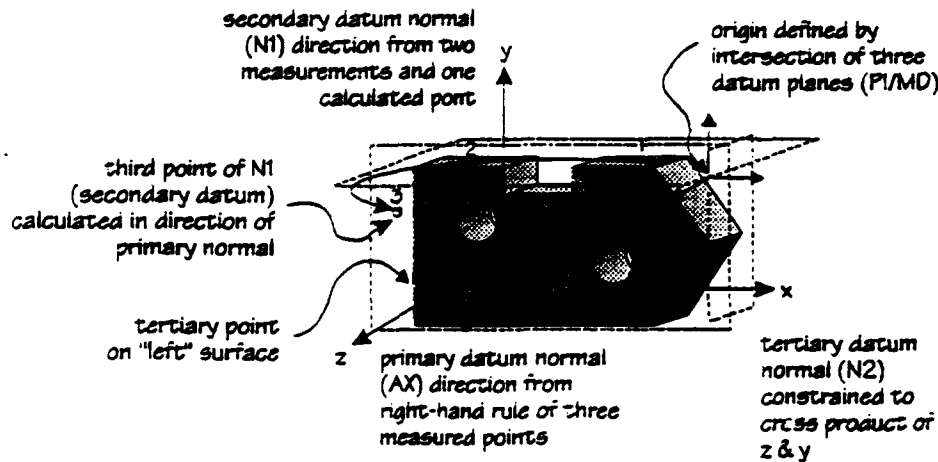


**Figure 38.** DRF creation within the CMES language.

## 4.3.2. Collision-free Path Planning

Because of the expense of automated robot equipment, an obviously crucial task in any robot motion planning is collision avoidance. It is a difficult task for the

inspector to perform manually. It involves the creation of the intermediate points, called via points, relative to the obstacles that the probe trajectory must circumnavigate, which is usually obvious to the inspector. Via points break the intersecting path into sub-paths which divert the colliding path around the obstacle. Then coordinate values of the via points relative to the setup orientation and DRF origin must be calculated, which is very difficult for the inspector due to the trigonometry, rotation, and translation that is required. However, the point coordinate calculation is trivial for the computer, but the point creation or point placement is difficult. Several obstacle avoidance algorithms have been proposed that deal with safe and efficient robot path planning. The algorithms can be grouped into the following classes:

1. hypothesize and test [Gewali, et al., 1990] [Pratt, 1991] [Bonner and Kelley, 1990]

2. penalty function [Brady, 1982]

3. explicit free-space [Sharir, 1986] [Lozano-Perez, 1979 & 1987]

The hypothesize and test method is the earliest proposal for robot obstacle avoidance. Its technique is to create a path candidate between the initial and final points, and to test for possible collisions. If a collision would occur, then a hypothesis creation method is proposed to create a new path candidate to test. This "hypothesize and test" step is repeated until a safe path is found, or the method times out.

The penalty function defines a numerical value for all possible probe positions within its workspace. The function returns an infinite penalty for positions that would cause collisions, then sharply drops off as the distance from the part increases.
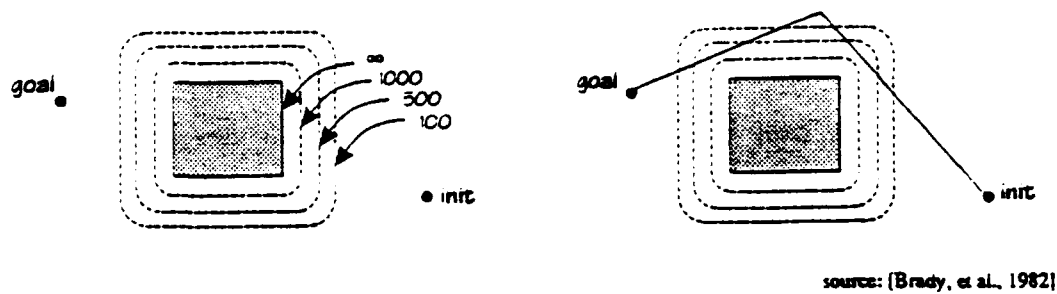
**Figure 39.** Penalty function to create a collision-free path.

The explicit free-space method builds explicit mathematical representations of subsets of robot configurations that are free of collisions.

The penalty function is attractive for combining the constraints from multiple objects; however, our problem consists of planning a path for one 3-D solid object. The free-space method works well in 2-D, but has been found to be computationally very expensive in 3-D [Bonner and Kelley, 1990]. The IPEM adapted the approach of the hypothesize and test method to discover a collision-free path. It was chosen for its simplicity (of both implementation and representation) and yet satisfactory solution for the problem encountered. However, an improved quality of solution and reduced computational time are developed by a number of facts and constraints that define this research project:

1. there exists just one obstacle (defining the CMM table as a constraint), the prismatic solid part model, in which to avoid probe collision;

2. there exists a calculable and reachable safe plane above and on all sides of the part within which the probe may always travel freely;

3. based on the inspection plan sequencing algorithm, all inspection points are accessible by the probe from the offset safe plane above the part; and

4. creating a DRF consists of measuring three mutually perpendicular surfaces.

Four different hypothesis test methods were implemented to utilize these facts. The first two methods do not actually perform a search through a defined space, but rather query the solid modeler about the geometric relationship between the initial surface and the goal surface. The third method does perform a search; however, it entails a reduced complexity by allowing the assertion that a collision-free path is always not more than two, or at the *most* three, sub-path combinations. Therefore, the search has a maximum limit of three sub-paths (two via points) to its search space from which it will return a failure. The fourth method not only creates a collision-free path without performing a search, but also does not query the solid modeler.
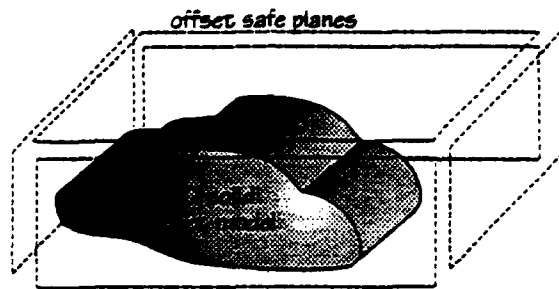


**Figure 40.** Safe planes are offset from the bounding box of the part-model.

The third and fourth methods of creating a hypothesis are adapted from the research of Gewali, et al., (1990) in path planning and 3-D vertical obstacles. Their research observes two propositions:

> **Lemma 1:** The shortest path [from point $s$ to $t$] in a $k$-story problem either a) lies completely in the base plane, or b) consists of three sub-paths of which the first of them rises from $s$ to some point $x_i$ on a horizontal obstacle edge at some level $L_i$, $0<i\leq k$, the second is a straight line segment connecting $x_i$ to $y_i$ on level $L_i$, and the third goes down from $y_i$ to $t$.

> **Lemma 2:** A shortest path from a point in the base plane $B$ to a point in level $L_i$ consists of straight line segments that form equal angles

with their projections on horizontal planes through their origin (i.e., these segments all slope upward at the same rate).

Since for any path over $L_k$, the offset safe plane can be projected onto $L_k$, no path will rise above it. In addition, a shortest path cannot rise above the base plane $B$ and return to it unless it has traveled over an obstacle. Otherwise, the projection of the path onto $B$ is guaranteed to be shorter and collision-free since the problem is constrained to a vertical polyhedron and has smaller and contained horizontal cross sections at higher levels, i.e., no tunnels or channels. It also follows that a falling section followed by a rising section can be projected onto the higher level and remain shorter and collision-free.
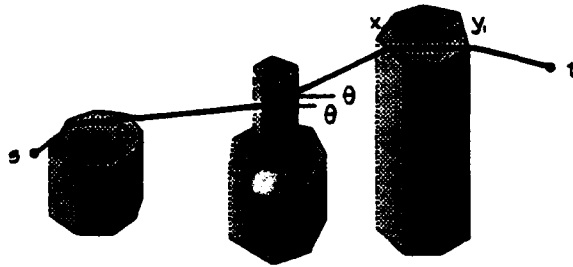


**Figure 41.** A maximum of three sub-paths needed to connect initial and goal points.

These same constraints are valid for the problems addressed in this research (section 2.2). The five degree of freedom (dof) probe is a goal of future work, but currently the three dof probe is considered. It was noticed from studying many examples of manually created CMM code from the QA engineers at the 4950th Test Wing, that the extra degrees of freedom above the standard three of the Cartesian robot motion were rarely used. Therefore, the inspection plan has no setups containing inspection points in tunnels or channels.

## Testing the Hypothesis

Testing the hypothesized path for collision required a unique implementation approach due to the limitations of the solid modeler, Shapes™ from XOX Corp. There are several solid (3-D) geometry Boolean commands (intersection, union, and difference) furnished by the solid modeler. The supplied intersection function specifies two solid geometry representations (geoms) as inputs and returns the new geometry resulting from the Boolean intersection operation. The computation time is large; however, more importantly, it is also unnecessary. The only answer needed to test the path hypothesis is "yes" or "no" — the fact of intersection — not the actual geometry resulting from the Boolean intersection! Determining the actual geometry becomes even more unreasonable due to the fact that one of the input geometries is the entire solid part model which is a complex, three-dimensional, feature-based object whose entire space would have to be searched.

The solution is a function that would return a value of "true" at the first finite element of detected intersection between the input geometries without testing any further to complete the resulting geometry. The function `intersection-p` was claimed by the solid modeler software manual to operate in this manner; however, it was discovered that it was actually just built upon the Boolean intersection function which calculates the entire intersection geometry structure, and just returns a "true" rather than the actual geometry.

The solution of this research was to implement a function to *efficiently* return a value of "true" if the proposed probe path intersects with the part model. The approach dissected the part-model into all of its 2-D surfaces. Then each surface was tested for intersection with the hypothesized path. If an intersection with a surface

would occur, the function stopped immediately and returned "true." This approach produced much faster results (on the order of three times, depending on when the intersection was found) since the intersection function only has to create an Boolean intersection geometry between the proposed path and a much smaller, less complex, two-dimensional surface geometry. The pseudocode of the function, called `check-for-intersection`, is as follows:

```
check-for-intersection:
    set x to all 2-D surfaces of the part-model
    for each line y in each sub-path in the proposed path do
        for each surface z in x do
            set r to result of the Boolean intersection of z and y
            if r is true, then
                quit from function and return true (intersection)
    return false (no intersection)
```

## Creating the Hypothesis

The implementation task of collision-free hypothesis forming can be defined as via point creation. The obstacles here are not polyhedron entities as used in [Gewali, et al., 1990], but are surfaces of the part-model. Movement around the surfaces is defined as passing around an edge; therefore, the via points are created from surface edges.

This research implemented four search methods to create the via points. These methods are called in a particular sequence by the collision avoidance algorithm. The sequence of methods starts with the simplest one. If the simplest method cannot find a path in its search criteria, the next method in the sequence is executed. Each subsequent method takes longer computationally than the previous method or its path is

less efficient. The final method is the "catch-all" method which is guaranteed to find a collision-free path, but does not take distance into consideration.

The hypothesis creation methods are named: neighbor, two-doors-down, 2-level-geometry, and up-and-over, which is also the sequence in which they are called. The collision-free path algorithm inquires each hypothesize and test method to create the collision-free path. If the proper information is not available or the method returns "false" (no path was found), the collision avoidance algorithm calls forth the next method in the sequence.

### Search #1: Neighbor Hypothesis Creation Method

The first method invoked is the neighbor search, and it is executed if both the initial and goal surfaces are known. It hypothesizes a collision-free path by searching for a common edge between the two surfaces signifying that they are neighbors. The surface which contains the initial inspection point is named *from-surface*, while the surface which contains the goal inspection point is named *to-surface*. If there exists a common edge between *from-surface* and *to-surface*, the via point is created as an off-set point from that common edge. The path connecting the initial point to the via point, and the via point to the goal point is tested to determine if it is collision-free using the check-for-intersection function described above. If no collisions are found, the via point is returned.
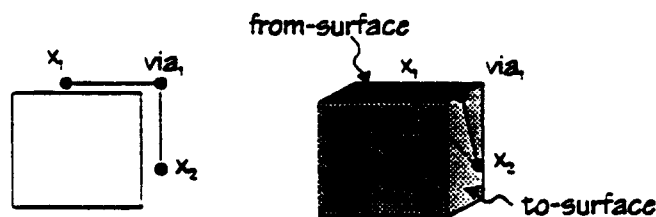


**Figure 42.** Neighbor hypothesis creation method.

The strength in this hypothesis creation method is two-fold. First, it is extremely simple and most via points are created using this method. Since GD&T requires the creation of three mutually perpendicular planes and many tolerances require more than one surface to be inspected, most collision avoidance routines simply move the probe around an edge from one surface to another, which is what the neighbor method explicitly queries about. The second strength is that the hypothesis creation method looks at geometry memory pointers or soft-pointers within the part-model geometry hierarchy which is very fast. The hypothesis creation does not actually create a path using the 3-D solid modeler, but queries the Concept Modeler™ for a geometrical relationship based on surface pointers stored in memory. The greatest advantage is that the computationally expensive collision testing does not occur if the geometrical relationship is not present. The pseudocode for the neighbor search is:

```
neighbor:
    if (to-surface and from-surface are known)
    and (to-surface ≠ from-surface), then
        set x to the Boolean intersection of the set of all edges from
            both to-surface and from-surface
        if x exits, then
            set via-point to (make-via-from-edge x p1 p2)
            if check-for-intersection of new path returns
             true, then
                return via-point
            else return false
```

The make-via-from-edge function returns a new point that will guide the path around the edge geometry it is given as its input. The via point is first created on the edge and then moved an offset distance from the edge in the directions of the normals of each surface of the edge. Its pseudocode is:

```
make-via-from-edge (x p1 p2):
    set via-point to the midpoint of the edge line x
    set n1 and n2 to normals of surface1 and surface2, respectively
    move via-point constrained to the direction along the edge line x to
        the average of p1 and p2.
    move via-point in direction for n1 an offset distance
    move via-point in direction for n2 an offset distance
    return via-point
```
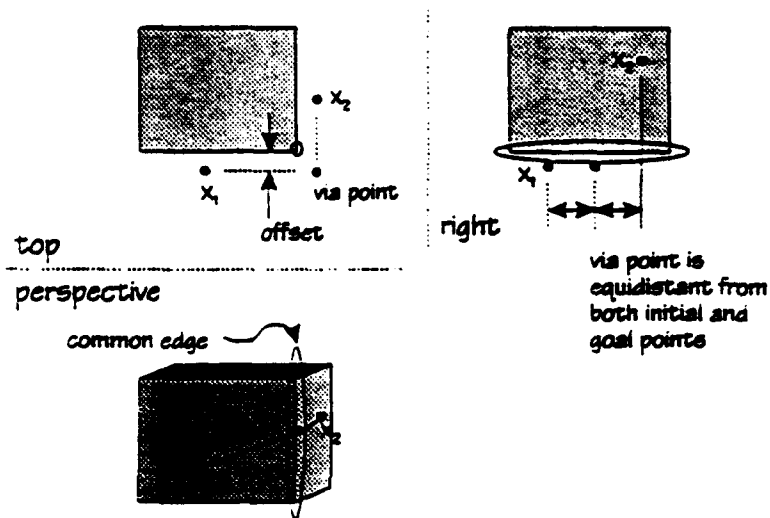


**Figure 43.** Via point creation from the `make-via-from-edge` function.

## Search #2: Two-doors-down Hypothesis Creation Method

If the neighbor hypothesis creation method failed to find a via point, then the collision avoidance algorithm calls forth the two-doors-down search. It hypothesizes a collision-free path by checking for a single surface that both input surfaces have as a neighbor. To describe their relationship as a cliché, *from-surface* and *to-surface* are "two doors down" or "my neighbor's neighbor." This is the next level of solid modeler query from the previously called neighbor search. If this intermediate surface is found, then two via points are created, one from the edge connecting *to-sur-*

*face* to the intermediate surface and one from the edge connecting the intermediate
surface to *from-surface*. If this hypothesized path has no collisions, then the two via
points are returned.
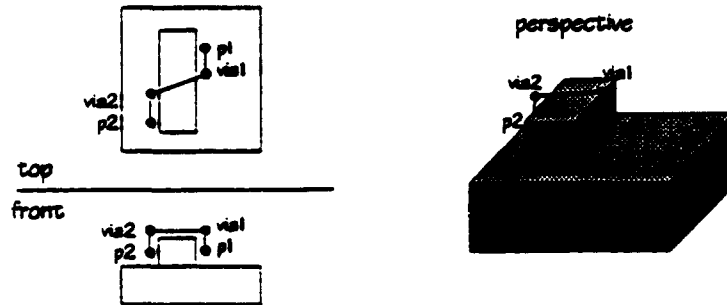


**Figure 44.** Two-doors-down hypothesis creation method.

The two-doors-down check is performed by comparing the "superior-geoms"
of the edges of both the initial and goal surfaces. The superior-geoms to an edge line
are the surfaces that create that edge, also described as the surfaces that the edge con-
nects together. Therefore, the Boolean intersection of the set of surfaces (superior-
geoms) of the edges of *to-surface* and *from-surface* will result in a common interme-
diate surface to which both are neighbors. The pseudocode is:

```
two-doors-down:
    for each edge x of to-surface do
        for each edge y of from-surface do
            set z to the Boolean intersection of the superior geoms of x
                and superior geoms of y
            delete to-surface and from-surface from z
            if z is not empty, then
                set i to point on z between p1 and p2
                set via1 to (make-via-from-edge x p1 i)
                set via2 to (make-via-from-edge y i p2)
                if total proposed path is collision-free, then
                    return via1 and via2
```

Notice that again, this hypothesis creation method does not loop through a search blindly looking for a collision-free path, but rather performs a computationally efficient query that requires looking at a memory look-up table.

### Search #3: 2-level-geometry-1 Hypothesis Creation Method

If the two-doors-down search fails to find a collision-free solution, the collision avoidance algorithm invokes the 2-level-geometry-1 method. This method got its weird name because it searches through two levels of geometry (2-level-geometry...) but requires only the initial surface to be known (...-1). Its approach is more like a true hypothesize and test search algorithm. However, since it is known that a collision-free path is always obtainable with only three sub-paths (two via points), this method will terminate its search and return failed if a path is not found after two levels of search.
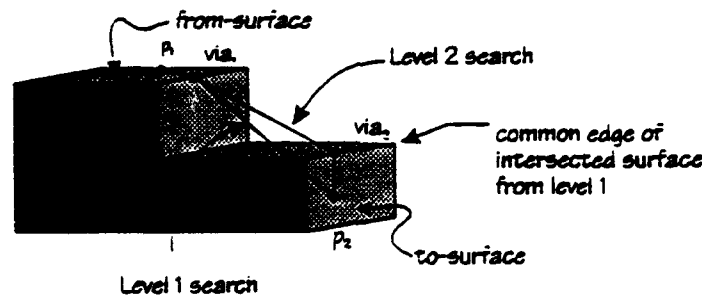


**Figure 45.** 2-level-geometry-1 hypothesis creation method.

The 2-level-geometry-1 method loops through each edge of *from-surface*, creates a via point, and hypothesizes a path from the initial point, to the via point, and then to the goal point (level 1). If the proposed path is collision-free, then the via point is returned; otherwise, the surfaces intersected are tested for their relationship. If the surfaces intersected by the level 1 hypothesis path have a common edge, then

that common edge will supply the other via point (level 2) and create the new hypothesized path. If the path containing the two via points search is tested false for collision detection, then the two via points are returned. The pseudocode is:

```
2-level-geometry-1:
    set x to the edges of from-surface
    for each edge y (a sub-geom) of x do
        set v₁ to (make-via-from-edge y p1 p2)
        set z to set of surfaces intersected by path from p₁ to v₁ to p₂
        if z is empty, then
            return v₁
        if (length of z = 2) and (there exists a common edge to surfaces
          in z), then
            set v₂ to (make-via-from-edge y v1 p2)
            set p to the path from p₁ to v₁ to v₂ to p₂
            if (check-for-intersection p) returns true, then
                return v₁ and v₂
```

### Search #4: Up-and-over Hypothesis Creation Method

The final method called forth by the collision avoidance algorithm is the up-and-over method, which is guaranteed to produce a collision-free path based on the utilization of how the inspection points are placed in a setup orientation. As discussed in section 4.1.3, an MR is placed into a setup object only if all inspection points pass the visibility and accessibility criteria. This ensures that the probe can safely reach the inspection point from above the offset safe plane. By definition of the offset plane, the probe may travel safely throughout the entire plane. Therefore, a collision-free path can be defined from an initial point, "up" to the offset plane, travel within the offset plane directly above the goal point, and then descend "down" directly to it.
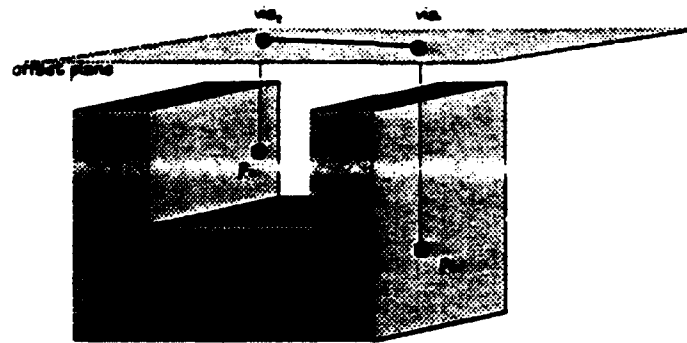
**Figure 46.** Up-and-over hypothesis creation method.

The up-and-over hypothesis creation method, as is the case for the first two methods, does not perform a looping search of possible paths, but it neither inquires about part-model geometry nor checks its path for collisions. It is called only after all sensible attempts to determine a collision-free path have already been executed. It performs the function of satisfying the primary and critical requirement of inspection path planning: to establish a collision-free path.

```
up-and-over:
    set v1 to a copy of p1
    move v1 into offset safe plane (directly "up")
    set v2 to a copy of p2
    move v2 into offset plane (directly "up")
    return v1 and v2
```

## 4.3.3. Plan Simulation

The popularity of process plan simulators for inspection and manufacturing re-iterates the importance of collision-free paths. Interpreting the code and displaying its results allows the inspector to foresee potential problems, inaccuracies, and overall efficiency. This capability is especially important due to the difficulty of interpreting

CMM code (even greater so for automatically generated CMM code) and the exorbitant cost of replacing the probe head, if damage should occur in a collision.

The IPEM provides a CMM path simulator in which the trajectory of the probe is overlaid on top of the part-model. This research also allows for collision detection within the simulator, with the colliding paths highlighted in red. This technique provides a double check to the collision avoidance algorithm as an assurance that the code is safe.

The format for the simulator is also aesthetically pleasing to the engineer. The RDS part-model display window is divided into four sub-windows, each containing a different view of the part-model and the overlaid probe path. The layout resembles an engineering drawing with top, front, and side views, but also includes a 3-D isometric view. A text window displays the current inspection plan information, such as setup number, DRF description, and feature/tolerance combination.
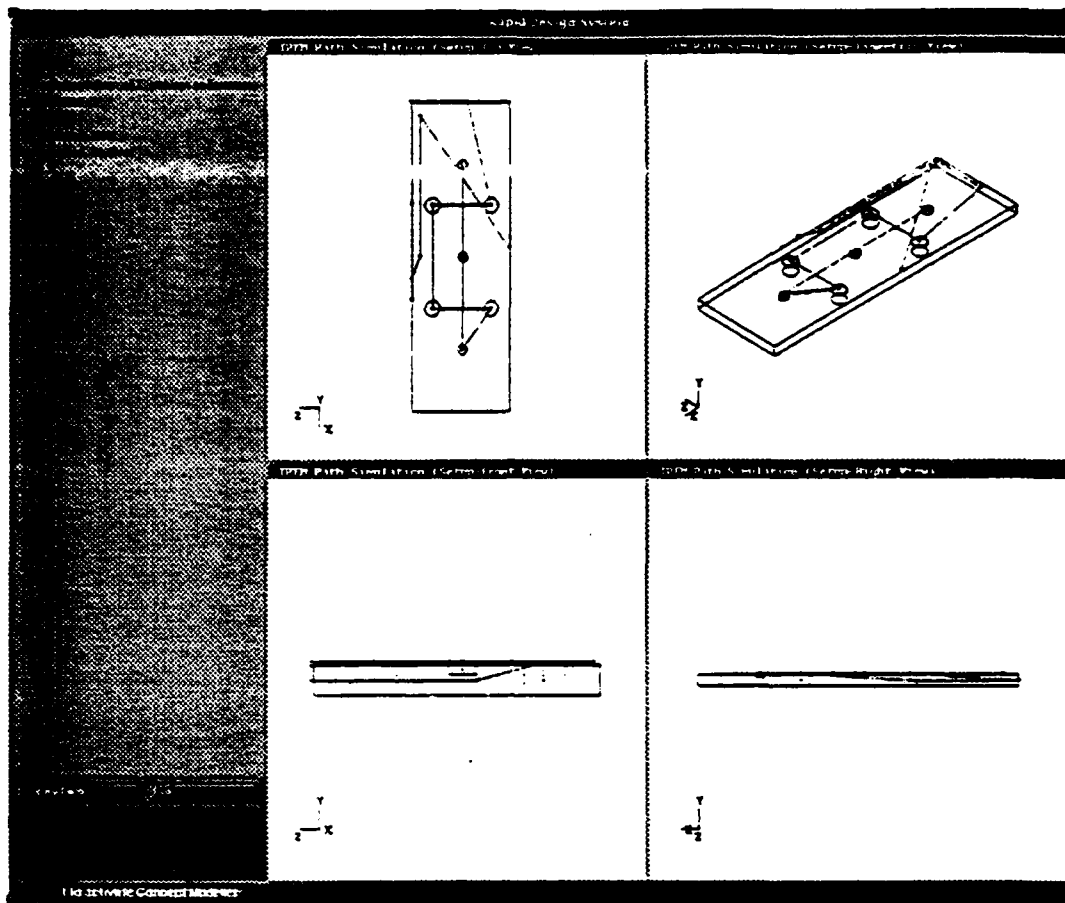
Figure 47. The simulation layout within the RDS.

## 4.3.4. Plan Translation

This thesis has brought the inspection planning process to its final step — generating the output. The desired output format is computer-controlled CMM instruction code for automated inspection. The current state of the inspection plan is represented by sequenced setup-DRF-MR structures consisting of optimized and collision-free measurement points. Plan translation between these two states has been implemented as a two-step process: translate the setup-DRF-MR structures into a generic

plan representation called "metacode," and then translate the metacode into the CMM code. The metacode is generated for several reasons:

1. the inspection plan is represented by the metacode in a "friendly" or conducive format for CMM code generation, whatever the particular CMM language;

2. the inspection plan in metacode representation may be saved and retrieved from disk;

3. the metacode structure can be easily altered to suit the needs of the inspector, inspection machines, and inspection techniques as they change, without requiring a total rework of the IPEM; and

4. the text and numerical values of the metacode can be easily edited by the inspector for quick code regeneration, rather than destroying and reinstantiating new whole new setup-DRF-MR structures.

The metacode received its name because it is a code that describes another code, i.e., the desired CMM code. The metacode representation removes the internal memory pointers to feature and geometry instantiations (geoms) from the inspection plan and replaces them with generic values in the form of text or numbers, which allows the storage and retrieval of the plan for future editing and re-generation by the inspector. Minor editing on the metacode (coined "tweaking") is probably unavoidable for most inspection plans, including such things as fixture avoidance, intelligent point placement among intersection features, and extra probe degrees of freedoms which are not yet implemented in the current research. Without metacode representation, the entire inspection process planning would have to start anew from the FBDE every time an old plan needed revision.

## MR to Metacode Translation

The metacode is created from the MRs in the setup-DRF-MR inspection plan representation generated by the process planner (section 4.1). The key to the efficiency of this feature translation is the object oriented software of CLOS (Common LISP Operating System). In CLOS, methods are defined based on the class definition of the argument passed. As a quick example, consider a software control for the hydraulic pumps in a mechanic's shop. Many methods named raise-car (they all have the same defined method name) would be defined describing what motor torque and height to use when raising a car based on the type of car class that was used as the method's parameter. Therefore, the proper raise-car method will be executed based on the class of its input argument, whether the car is an Integra or an Impala.

The MR to metacode translation algorithm calls a method named generate-metacode-4this-tol&feat and passes an MR instantiation. There are many different methods with this name, each of which is defined to recognize an MR object by the class of its tolerance feature *and* its design feature. For example, a call with a position tolerance and hole feature will execute a different method definition than a call with a concentricity tolerance and hole feature. Defining a method based on two or more classes is described further by Keene (1989).

This translation is not a 1:1 ratio. The MR is designed to efficiently represent one aspect of the inspection plan — the measurements. The metacode is designed to efficiently represent the tolerance evaluation. The metacode consists of three types of measurement and evaluate commands:

1. measure and evaluate;

2. measure, evaluate, and save the measurement values for later evaluation; and

3. use saved points from previously saved measurements to perform a different evaluation function.

One process involved in the MR translation to metacode is that the inspection points are translated into their proper setup-DRF orientation. Within the MR, all inspection points are created with reference to the internal origin of the CAD system, which is defined to be the center of mass of the part-model. GD&T requires the origin of the measurements to be at the intersection of the three datum planes.

Before translation of the inspection points, the DRF origins must be calculated with respect to the internal CAD origin. This defines the translation vector needed to move each point into the DRF reference frame. However, when axis datums are introduced, the planes of the coordinate frame become slightly obscure since an axis is contained by an infinite number of planes. Therefore, inspection algorithms are used to define the directions of the coordinate frame planes.
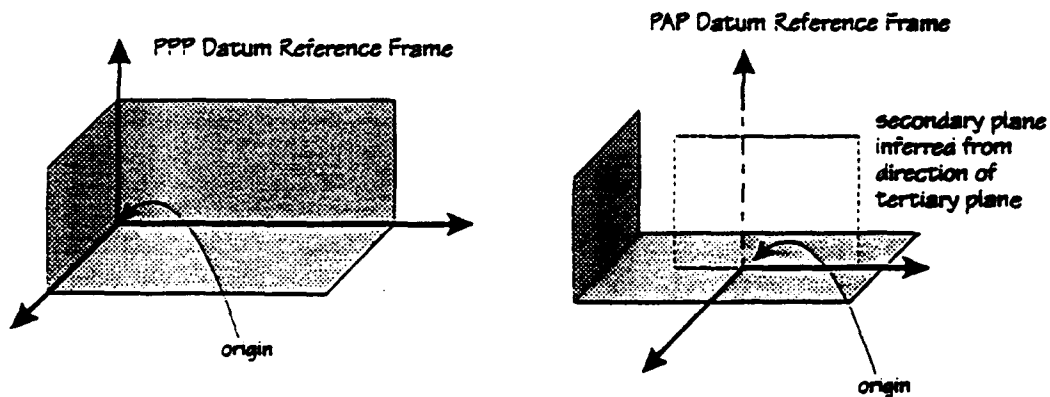


Figure 48. Origin and axes creation defined from the datum of the DRF.

Once the DRF origins are formed, all the inspection points and offset points must be translated and rotated into the proper setup-DRF coordinate frame from the internal CAD coordinate frame. The standard robotics $A$ matrix is used for this conversion.

$$A = \left[ \begin{array}{ccc|c} & ^{CAD}R_{setup} & & x_0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \qquad (4.15)$$

where $^{CAD}R_{setup}$ is defined as $R(\phi,\theta,\psi)$ in (4.1) which rotates the coordinate frame from the CAD representation into the setup orientation, and $x_0$ is the DRF translation vector. Figure 49 shows the many coordinate transformations involved in the inspection planning process.
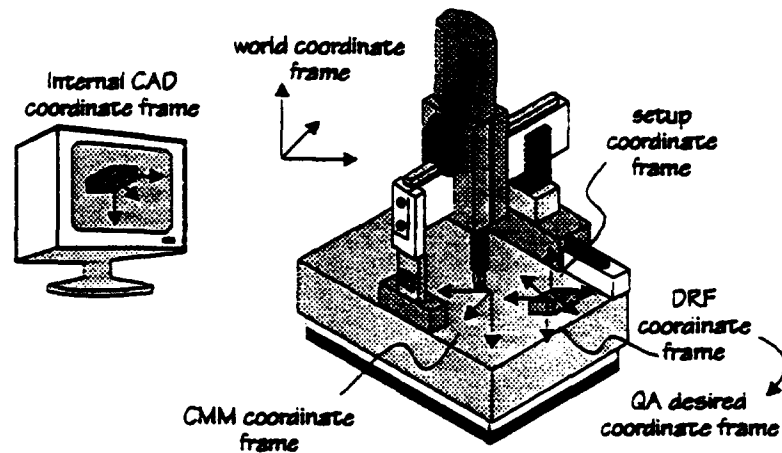


**Figure 49.** The coordinate frames within the inspection scheme.

Finally, the MR to metacode translation also calculates special information needed by pre-defined CMM measurement algorithms. For example, to inspect a hole feature, most CMM languages have internal commands to do this, which only

need the theoretical center of the hole and the diameter. The CMM will automati-
cally move to the center and probe three or four points at 120° or 90° apart, respec-
tively. The MR does not contain this information, but rather actual inspection points
on the inside surface of the hole, which is inefficient for most automated CMM pro-
gramming.

## 4.3.5. Code Generation

The automated generation of the CMM code is simple due to the structure of
the metacode and the ready-to-use information it contains. Any number of code
generators can be called by the inspector to generate the desired output format.
Currently, CMES is the only supported code generation format, but future implemen-
tations will include DMIS and manual inspection representations.

### Implementation

Since this thesis is concerned with using the inspection rule based ANN to
sequence the inspection points (section 4.2.2), the CMM code generator follows the
plan structure of Figure 29(c) on page 55. Macros are filled with variables from the
metacode data slots to output the code in the proper format. Appendix A shows the
macros used to output the CMES codes from the design feature/tolerance feature
combinations stored in the metacode. The IPEM also has the ability to output the
plan structure of Figure 29(a), if the two-level nearest neighbor search is used to
sequence the inspection points (section 4.2.1).

One final feature is that the code generator can also perform functions to con-
figure the CMM code into a particular appearance desired by the inspector. This re-
search added another inspection point transformation at the request of the end-user.

the QA engineers at the Air Force 4950th Test Wing, who desired a consistency in the coordinate frame for all setups. Even when the product rotates on the table, the QA engineers want the x-y-z axes to still point in the same direction. Their request was for z to point "up," y to point "left," and x to point "back." To accomplish this configuration, the setup orientation matrix is used backwards to rotate the coordinate frame from the perspective of the part-model, rather than the part-model and coordinate system from the perspective of the viewer.

$$
A = \begin{bmatrix} & & & \vdots & 0 \\ & R_{xyz'}(\psi,\theta,\phi) & & \vdots & 0 \\ & & & \vdots & 0 \\ \hline 0 & 0 & 0 & \vdots & 1 \end{bmatrix}
\qquad (4.16)
$$

The result puts the product into the setup orientation with the coordinate frame matching the internal CAD coordinate frame. Since the internal CAD orientation and QA desired orientation are both constants, a standard frame rotation allows the transformation between them.

$$
A = \begin{bmatrix} & & & \vdots & 0 \\ & R_{xz'}(90,-90) & & \vdots & 0 \\ & & & \vdots & 0 \\ \hline 0 & 0 & 0 & \vdots & 1 \end{bmatrix}
\qquad (4.17)
$$

CAD default        part in setup orientation,        part in setup orientation,
                         frame in CAD default             frame in QC desired
                                                      orientation
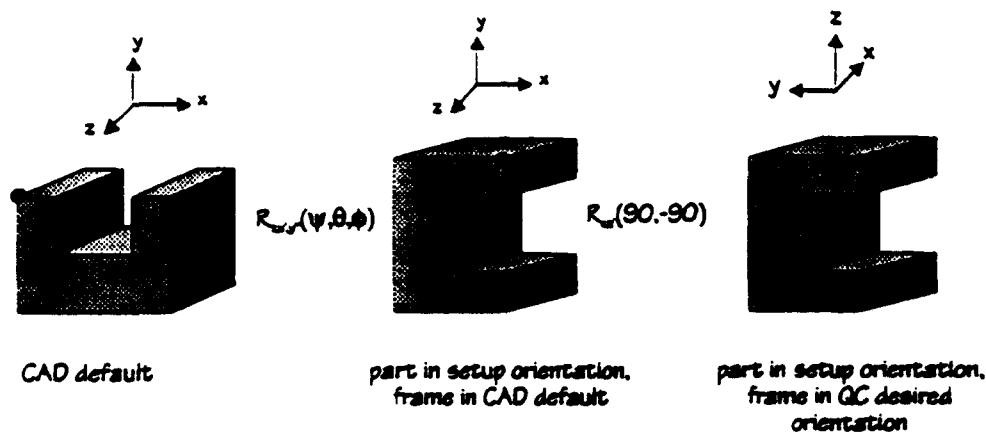
**Figure 50.** Part-model and axes rotation from CAD default to inspector desired orientation.

The automatically generated CMM code that evaluates the tolerances on the

part-model simulated in Figure 47 on page 95 is shown below:

```
CMES program to inspect the part          DA
"boomer panel"                            \\\\\\\\
Created on 12:37:37 7/11/1992.              Move probe clear of obstructions
                                          \\\\\\\\
Description: boomer panel for fuel refill   Enter TI when ready to continue
Inspector: stever                         \\\\\\\\
                                          ! Paused to wait for operator response
Notes to instructor for setup:            ET
1. Aligned the part with the primary datum ! Re-Index PH10 Probe Head
   surface on the cmm table. the          UR.1.PH
   secondary datum facing the inspector   ! Continue Inpection Program
   (computer), and the tertiary datum     \\\\\\\\
   facing to the left (the door)            Take 3 touches on the TOP of the TABLE
                                            SURFACE -z- CCW
   TI to continue                         AX-Z

ET                                          Take 2 touches on the FRONT surface,
RC\NP\EP,LP:\SU\N                            RIGHT FIRST
PA,LP:\PA                                  N1-X,Z
OP\DT
=================================           Take 1 touch on the LEFT end of the part
   Setup #1                               N2-,Y,X,Z
! Prompt operator to clear PH 10 probe    P\MD\SA.2
```

\\\\\\\\
MOVE PROBE CLEAR OF
OBSTRUCTIONS, TI TO CONTINUE
ET

(DTM-1 A DRF-1 P) with Datum
tolerance in DRF-1
setup #1, CMES #1
———————————————

#MC\-14.130\-9.772\0.100
#PT,Z\0.000
#MC\-14.130\-9.772\0.100

#MC\-1.271\-0.854\0.100
#PT,Z\0.000
#MC\-1.271\-0.854\0.100

#MC\-14.995\-9.151\0.100
#PT,Z\0.000
#MC\-14.995\-9.151\0.100

AX-,Z

(DTM-2 B DRF-1 S) with Datum
tolerance in DRF-1
setup #1, CMES #2
———————————————

#MC\-19.108\0.100\-0.016
#PT,Y\0.000
#MC\-19.108\0.100\-0.016

#MC\-9.870\0.100\-0.500
#PT,Y\0.000
#MC\-9.870\0.100\-0.500

N1-,Y,Z

(DTM-3 C DRF-1 R) with Datum
tolerance in DRF-1
setup #1, CMES #3
———————————————

#MC\0.100\-5.509\-0.500
#PT,X\0.000
#MC\0.100\-5.509\-0.500

N2-,X,Y,Z

PNMD\SA,3

DRF-1 creates a new origin to which all
points from now on refer.
══════════════════════════════════

via-point(s)
——

#MC\0.100\-5.509\0.100
#MC\-10.000\-8.000\0.100

TH-6 with Position tolerance in DRF-1
setup #1, CMES #9
———————————————

#ID,Z//\-10.000\-8.000\
-0.500\1.500,0.01,0.01\5.0E-4
#MC,Z\0.100

TH-2 with Position tolerance in DRF-1
setup #1, CMES #7
———————————————

#ID,Z//\-10.000\-2.000\
-0.500\1.500,0.01,0.01\5.0E-4
#MC,Z\0.100

TH-3 with Position tolerance in DRF-1
setup #1, CMES #6
———————————————

#ID,Z//\-20.000\-2.000\
-0.500\1.500,0.01,0.01\5.0E-4
#MC,Z\0.100

TH-7 with Position tolerance in DRF-1
setup #1, CMES #4
———————————————

#ID,Z//\-20.000\-8.000\
-0.500\1.500,0.01,0.01\5.0E-4
#MC,Z\0.100

BH-2 with Position tolerance in DRF-1
setup #1, CMES #5
———————————————

#ID,Z//\-24.000\-5.000\
-0.150\1.000,0.01,0.01\0.01
#MC,Z\0.100

BH-1 with Position tolerance in DRF-1
setup #1, CMES #8
———————————————

#ID,Z//\-15.000\-5.000\
-0.150\1.000,0.01,0.01\0.01
#MC,Z\0.100

BH-3 with Position tolerance in DRF-1

setup #1, CMES #10

#ID,Z///\-6.000\-5.000\
-0.150\1.000,0.01,0.01\0.01
#MC,Z\0.100

# Chapter 5

# Results

---

This chapter will re-illustrate the performance of the IPEM by showing all aspects of the intelligent inspection system. A single product design will be taken from CAD creation, through the IPEM, and finally to executable CMM code. Monitor snapshots of the process will show both internal representations as well as the human interfaces, and actual text results will be displayed.

The product is created in the FBDE, the feature-driven CAD module of the RDS (section 2.1). For this part-model, the designer added negative-volume features to a positive-volume rectangular starting block. Two blind-hole features and one pocket feature are attached to the starting block, and another two blind-hole features are attached to the bottom of the pocket.
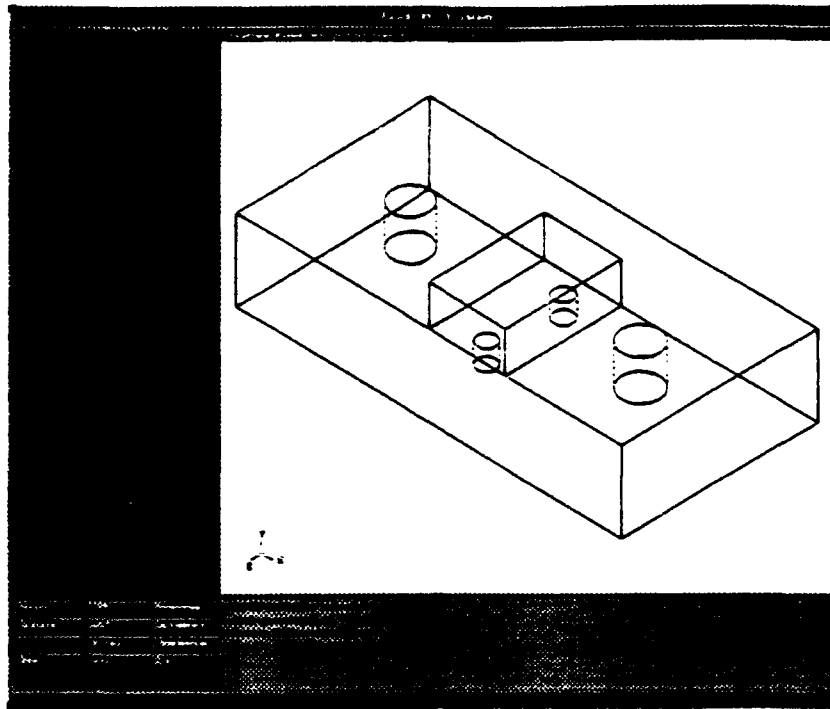
**Figure 51.** The Feature Based Design Environment.

The designer also places the GD&T callouts onto the part-model by attaching them to features or sub-features, i.e., surfaces of features. Figure 52 shows the tolerance callouts of the part-model. Since the FBDE does not currently display the tolerance, GD&T feature control frames were graphically overlaid on top of the snapshot of Figure 51.
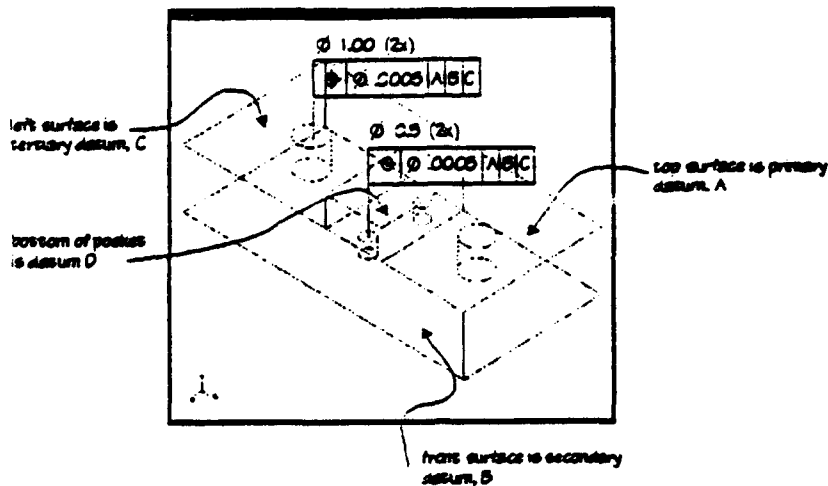
**Figure 52.** Tolerances overlaid on part-model.

Assuming that the product has been fabricated by the manufacture engineers using the FAB-PLAN module of the RDS, the workpiece is given to the QA engineers who locate the part-model in the RDS database and bring it into the IPEM. The IPEM has the same window layout as the FBDE (Figure 51); therefore, Figure 53 shows the new button menu of the IPEM layout.
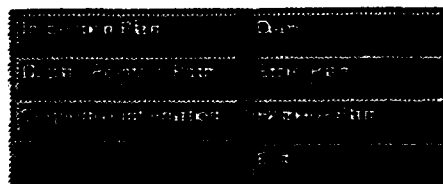


**Figure 53.** The button menu of the Inspection Planning and Evaluation Module layout.

To start the automated inspection planning, the **Inspection Plan** button is clicked on by the inspector (Figure 53). The IPEM begins to execute through all the steps outlined in section 4.1: stable surface location, setup creation, MR creation, and

setup population. Messages displayed to the inspector indicate the currently active step. Other than that, nothing else is displayed due to the large increase in execution time that entails displaying the geometries.

Once the process planner has represented the inspection plan by the setup-DRF-MR structures, the schedule optimization begins by generating the three inspection rule matrices:

**W matrix:**

```
0.000 1.000 0.632 0.671 0.281 0.614 0.700 0.571
1.000 0.000 0.407 0.407 0.736 0.463 0.330 0.463
0.632 0.407 0.000 0.245 0.407 0.253 0.137 0.061
0.671 0.407 0.245 0.000 0.407 0.061 0.137 0.253
0.281 0.736 0.407 0.407 0.000 0.352 0.447 0.352
0.614 0.463 0.253 0.061 0.352 0.000 0.173 0.245
0.700 0.330 0.137 0.137 0.447 0.173 0.000 0.173
0.571 0.463 0.061 0.253 0.352 0.245 0.173 0.000
```

**F matrix:**

```
-0.667  0.333  1.000  1.000  0.333  1.000  1.000  1.000
 0.667 -1.000  0.000  0.000 -1.000  0.667  0.667  0.667
 0.667  0.333 -0.667 -0.667  0.333  0.667  0.667  0.667
 0.667  0.333 -0.667 -0.667  0.333  0.667  0.667  0.667
 0.667 -1.000  0.000  0.000 -1.000  0.667  0.667  0.667
 0.667  1.000  0.667  0.667  1.000 -0.667 -0.667 -0.667
 0.667  1.000  0.667  0.667  1.000 -0.667 -0.667 -0.667
 0.667  1.000  0.667  0.667  1.000 -0.667 -0.667 -0.667
```

**S matrix:**

```
1 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0
0 0 0 1 1 1 1 1
0 0 0 1 1 1 1 1
0 1 1 0 0 0 0 0
0 0 0 1 1 1 1 1
0 0 0 1 1 1 1 1
0 0 0 1 1 1 1 1
```

See Figure 36 on page 73 for a review of what the matrices represent. The IPEM then makes a call to the ANN program (Appendix B) to execute the optimization scheduling using inspection rules on the measurement points. The resultant path trajectory is then displayed to the screen. Figure 54 shows the RDS display-window;

however, sequence numbers were overlaid onto the snapshot to help clarify the path

trajectory to the reader.
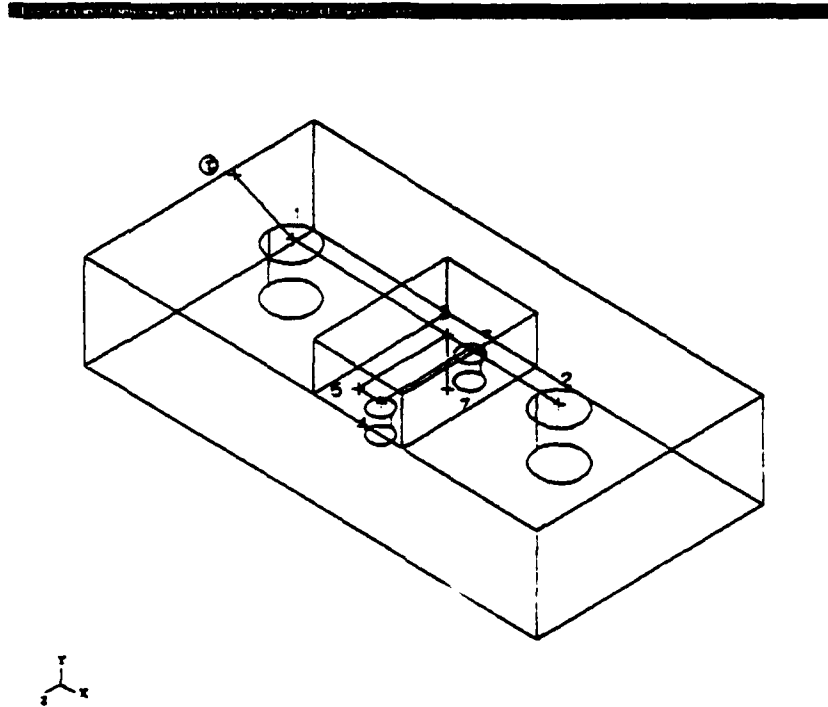


**Figure 54.** Results from artificial neural network schedule optimization.

This path represents adherence to the inspection rules described in section 4.2.2. The

first two blind holes measured have a tolerance value smaller than the inspector-de-

fined "tight" tolerance; therefore, these measurements should go before any other

tolerance in the inspection plan according to the tightness inspection rule. The ANN

was driven to this solution by the tightness rule influencing the $S$ and $F$ matrices.

The two blind holes attached to the bottom of the pocket are measured next because

they are of the same feature type as the previous blind holes, as required by the

feature inspection rule and implemented into the $F$ matrix. Finally, the three

remaining inspection points are measured according to a minimum Euclidean distance criterion implemented in the $W$ matrix.

At this point, the optimized path intersects the part-model. The collision avoidance algorithm is now executed; it first asks if user interaction breaks are desired within the algorithm. These breaks allow the inspector to accept a proposed path, or reject it by telling the algorithm to keep searching for a different collision-free path. This is useful in the 2-level-geometry-1 search where one edge might produce an acceptable collision-free path, but the inspector can clearly see that a yet untested edge will produce a better path. Figure 55 shows the collision avoidance algorithm results, again with text overlaid on the graphic to clarify the paths to the reader.
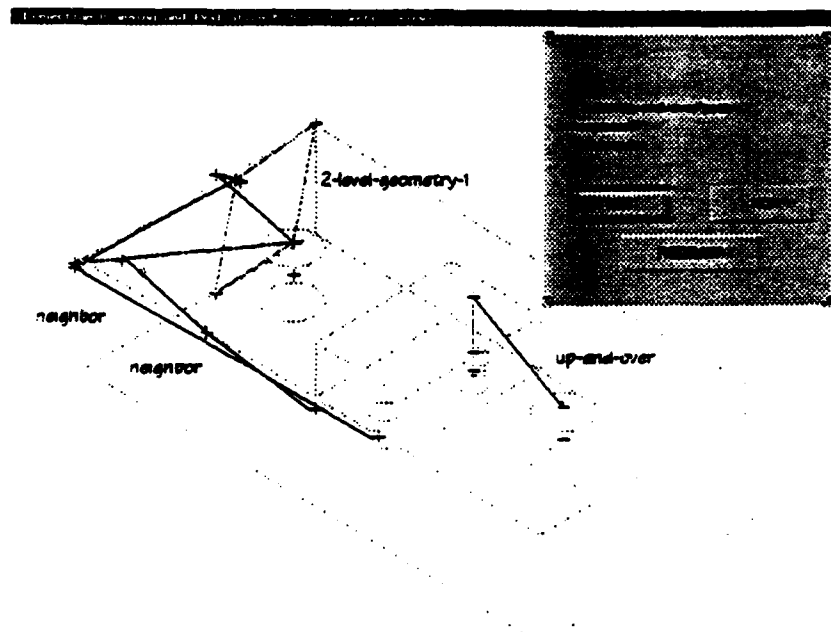


**Figure 55.** Collision avoidance algorithm.

The MR representation of the inspection plan, with the inspection points optimized and the via points created, is sent to the automated CMM code generator. First the MR representation is translated into metacode representation. A "pretty-printer" output of the metacode is shown below. The : and ... symbols represent truncated code, deleted due to redundancy, since all metacode instantiations have the same slot structure.

```
Inspection Plan:
Inspector: "stever"
Part name: "path-bar"
===============
|setup-number: 1
|normal-of-resting-surface: (0.0 1.0 0.0)
|orientation-matrix: (0.0 0.0 0.0)
|drf-lists: 1 total
|---------------
|* New DRF
|---------------
||drf-number: 1
||drf-name: DRF-1
||drf-label: ABC
||drf-type: PPP
||primary-datum:
||_
|||feature-type: PLANE
|||tolerance-type: DTM
|||drf: DRF-1
|||approach-vector: 3
|||pts-list: ((3.6296 -1.0 1.0832) ...)
|||offset-pts-list: ((3.6293 -1.1 1.0832)
|||cmm-pts-list: (...)
|||cmm-offset-pts-list: (...)
|||tol-value: P
|||mat-cond: NIL
|||num-points: 3
|||feature-name: (DTM-1 A DRF-1 P)
|||feature-dims: NIL
|||save-meas?: NIL
|||setup-number: 1
|||drf-number: 1
|||mr-number: 1
|||surface: 268439487
|||via-points: NIL
|||via-method: NIL
```

```
||-
||secondary-datum:
||    :
||_
||-
||tertiary-datum:
||    :
||_
||metacode-lists: 5 total
||_
|||feature-type: HOLE
|||tolerance-type: PO
|||drf: DRF-1
|||approach-vector: 2
|||pts-list: ((30. 0.5 0.0))
|||offset-pts-list: ((3.0 1.1 0.0))
|||cmm-pts-list: ((-8.0 -2.5 1.5))
|||cmm-offset-pts-list: ((-8.0 -2.5 2.1))
|||tol-value: 1.0E-5
|||mat-cond: NONE
|||num-points: 3
|||feature-name: BH-2
|||feature-dims: (1.0 1.0)
|||save-meas?: NIL
|||setup-number: 1
|||drf-number: 1
|||mr-number: 4
|||surface: 268439269
|||via-points: NIL
|||cmm-via-points: NIL
|||via-method: NIL
||-
||_
|||feature-type: HOLE
|||tolerance-type: PO
|||    :
|||feature-name: BH-3
```

```
III     :                              IIIfeature-type: T
II-                                     IIItolerance-type: DTM
II_                                     III     :
IIIfeature-type: HOLE                   IIIfeature-name: PKT1[5]
IIItolerance-type: PO                   III     :
III     :                               II-
IIIfeature-name: BH-4                   IInn-via-points: ((-5.1 1.1 -.855)...)
III     :                               IInn-sequence-points: (3 0 7 0 4 0 5 0 6 0 8 ...)
II_                                      I-
IIIfeature-type: HOLE                    minmax-box: ((-5.0 5.0) (-1.0 1.0) (-2.5 2.5))
IIItolerance-type: PO                    offset: 0.1
III     :                                cmm-language: CMES
IIIfeature-name: BH-1                     cmm-accuracy: 0.00001
III     :                                 drf-origin-list: ((DRF-1 (-5.0 -0.1 2.5)))
II-                                       drf-axes-list: ((DRF-1 (3 4 1)))
II_
```

The CMES code is then generated from the metacode. The inspector is prompted for a filename into which the text is placed. This file will then be taken to the CMM computer controller either by diskette, modem, or network transfer. The program will be downloaded to the CMM language interpreter, CMES, and executed to inspect the workpiece.

```
CMES program to inspect the part "path-bar"    Inspector: stever
Created on 14:30:34 6/11/1992.                 ==============================
                                                        Setup #1
Description: bar part for placing NN path over
Inspector: stever                              ! Prompt operator to clear PH 10 probe
                                               DA
Notes to instructor for setup:                 \\\\\\\\
1. Aligned the part with the primary datum       Move probe clear of obstructions
   surface on the cmm table, the                \\\\\\\\
   secondary datum facing the inspector           Enter TI when ready to continue
   (computer), and the tertiary datum           \\\\\\\\
   facing to the left (the door)                 ! Paused to wait for operator response
                                               ET
   TI to continue                              ! Re-Index PH10 Probe Head
                                               UR.1.PH
ET                                             ! Continue Inpection Program
RC\NP\EP.LP:\SU\IN                             \\\\\\\\
PA.LP:\PA                                        Take 3 touches on the TOP of the TABLE
OP\DT                                            SURFACE -z- CCW
==============================                  AX-.Z
Part: path-bar
Description: bar part for placing NN path over
```

Take 2 touches on the FRONT surface.
RIGHT
FIRST
N1-,X,Z

Take 1 touch on the LEFT end of the part
N2-,Y,X,Z
P\MD\SA,2
\\\\\\\\
MOVE PROBE CLEAR OF
OBSTRUCTIONS, TI TO CONTINUE
ET
Re-creating danums for DRF-1

(DTM-1 A DRF-1 P) with Datum tolerance in
DRF-1
setup #1, CMES #1

---

#MC\1.417\-8.629\1.200
#PT,Z\1.100
#MC\1.417\-8.629\1.200

#MC\4.886\-4.710\1.200
#PT,Z\1.100
#MC\4.886\-4.710\1.200

#MC\0.427\-0.424\1.200
#PT,Z\1.100
#MC\0.427\-0.424\1.200

AX+,Z

via-point(s)
---
#MC\-0.100\-2.711\1.200

(DTM-2 B DRF-1 S) with Datum tolerance in
DRF-1
setup #1, CMES #2

---

#MC\-0.100\-4.998\0.887
#PT,X\0.000
#MC\-0.100\-4.998\0.887

#MC\-0.100\-6.369\1.069
#PT,X\0.000
#MC\-0.100\-6.369\1.069

N1+,X,Z

via-point(s)

---

#MC\-0.100\0.100\0.978

(DTM-3 C DRF-1 R) with Datum tolerance in
DRF-1
setup #1, CMES #3

---

#MC\3.355\0.100\0.887
#PT,Y\0.000
#MC\3.355\0.100\0.887

N2-,Y,X,Z

P\MD\SA,3

DRF-1 creates a new origin to which all
points from now on refer.

===

via-point(s)
---
#MC\3.355\0.100\1.200
#MC\2.500\-2.000\1.200

BH-1 with Position tolerance in DRF-1
setup #1, CMES #7

---

#ID,Z//\2.500\
-2.000\0.600\1.000,0.01,0.01\1.0E-5
#MC,Z\1.200

BH-2 with Position tolerance in DRF-1
setup #1, CMES #4

---

#ID,Z//\2.500\
-8.000\0.600\1.000,0.01,0.01\1.0E-5
#MC,Z\1.200

via-point(s)
---
#MC\2.500\-8.000\1.200
#MC\3.500\-5.000\1.200

BH-3 with Position tolerance in DRF-1
setup #1, CMES #5

---

#ID,Z//\3.500\-5.000\
-0.150\0.500,0.01,0.01\0.01
#MC,Z\0.200

BH-4 with Position tolerance in DRF-1

```
setup #1. CMES #6                    #MC\1.500\-4.500\0.200
------------------------------
#ID.Z//\1.500\-5.000\               #MC\3.500\-4.500\0.200
-0.150\0.500.0.01,0.01\0.01          #PT.Z\0.100
#MC.Z\0.200                          #MC\3.500\-4.500\0.200

                                     #MC\2.500\-5.500\0.200
STARTING-BLOCK with Flatness tolerance  #PT.Z\0.100
in DRF-1                             #MC\2.500\-5.500\0.200
setup #1. CMES #8
------------------------------       FF. Z/PL\0.010\10.000
#MC\1.500\-4.500\0.200
#PT.Z\0.100
```

When the IPEM has completed the CMES code generation, the inspector can perform two aids provided by this research. First, the inspector can now save the metacode to disk. When the **Save Plan** button is clicked by the mouse, the inspector is prompted for a file name to place the metacode into. If the CMM code needs to be regenerated, or the metacode needs editing, the file can be retrieved from the storage file and save the inspector a lot of time rather than performing the process planning again. The **Retrieve Plan** button performs this function.

Secondly, the inspector can click on the **Display Points & Path** button to start the CMM simulation. This brings up the four-view engineering design layout and overlays the probe trajectory onto the part-model.
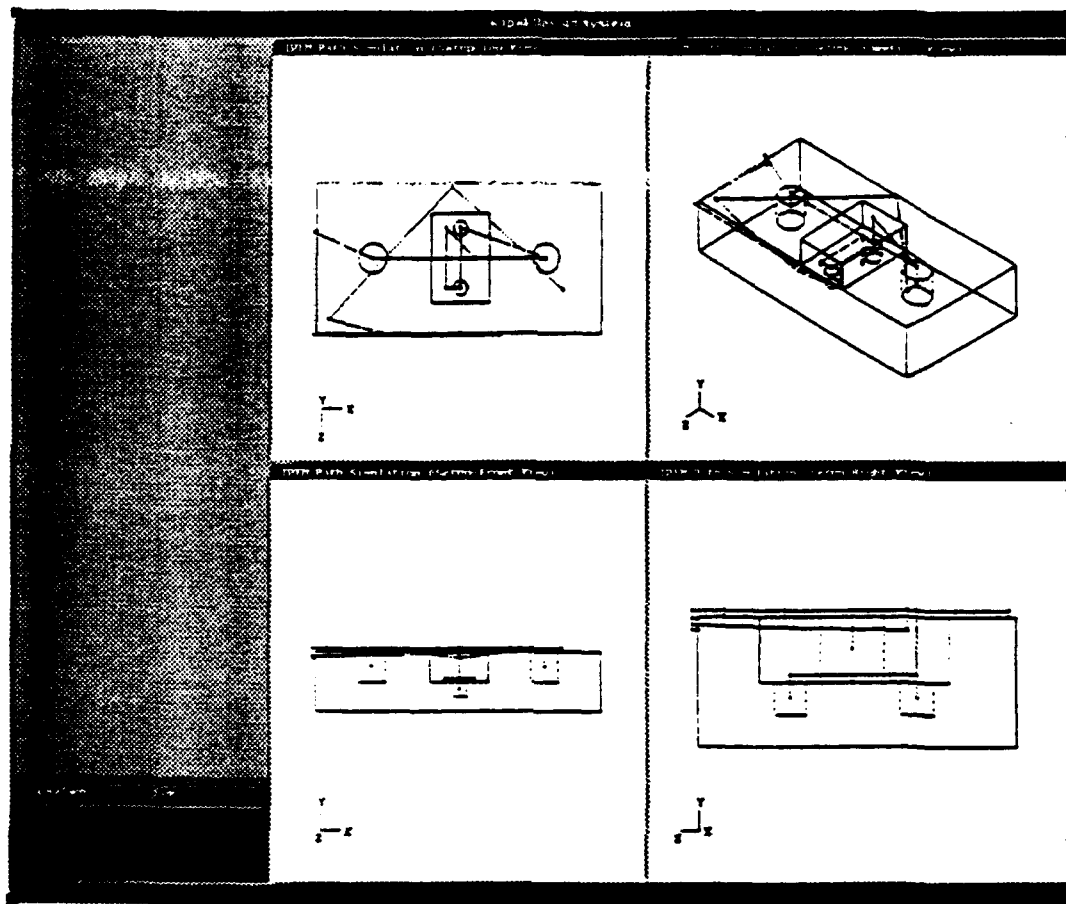
**Figure 56.** CMM simulation.

# Chapter 6

# Conclusions

*Research is to see what everyone else has seen,
and to think what nobody has thought.*
- Dr. Albert Szent-Cyöryi

The outcome of this research is a number of accomplishments seen from many different perspectives. Significant results have been produced in several areas of research.

Looking at the larger picture, this research has had a large part in producing a viable automated inspection planner that is integrated into the RDS, a feature-based concurrent engineering system. The implemented work of this thesis has converted the IPEM from a research idea only into an application that produces usable results and provides benefits to its users.

From the perspective of scheduling optimization, the use of two different forms of rule-based matrices to integrate rules into an artificial neural network is unprecedented. Three different rule-generated matrices were implemented into the neuron motion equation of the ANN. The output produced modeled the desired results and sequenced the inspection points using inspection rule criteria.

A new approach to producing collision-free paths was also introduced. The collision avoidance algorithm modified the generic hypothesize and test method into a computationally efficient algorithm based on an automated inspection paradigm.

116

Four different hypothesis creation methods were implemented, each with a quick search or part-model query for a typical geometrical relationship between the initial and goal points. If that relationship is found, then the method can efficiently produce the via points needed to create a collision-free path.

Simulating the plan before automated inspection ensures the inspector that the probe path is safe and efficient. The inspector can double check the output of the point scheduling and collision avoidance algorithms for desired results..

Representing the inspection plan in metacode format ensures an easy upgrade in the IPEM for future additions to output formats. The metacode also provides for storage and retrieval functions. Editing capabilities can be efficiently performed through the use of the metacode instantiations which do not contain internal memory structures.

The automatic generation of CMM code uses a metacode to CMES generator. The code is created from generic CMES macros that are populated with the proper calculated values. These macros can also be edited by the inspector to ensure an easy software upgrade and to customize the output of the inspection plan.

# Chapter 7

# Future Work

> *Our efforts today and what we have done so far*
> *are but building blocks in a huge pyramid to come ...*
>
> *Knowledge begets knowledge. The more I see,*
> *the more impressed I am — not with what we know — but*
> *with how tremendous the areas are that are as yet unexplored.*
> - Lt. Col. John H. Glenn, Jr.

The RDS is a large research project that has accomplished much, but has many new avenues to explore. With respect to inspection planning and this research, the future work entails improvements to existing techniques, as well as new implementations:

*Variant process planning*: variant planning would be in addition to the current generative process planning. Variant process planning begins with a previous inspection plan and updates it according to the current part-model and tolerances.

*Intelligent inspection point placement on toleranced surfaces*: currently the points are placed on the surfaces by a random process. An intelligent placement based on inspection rules or neural net learning would improve the evaluation results.

*Intersecting, interacting features*: an addition to the IPEM would provide intelligent point placement on a surface that is not complete because of feature interaction.

*Neural network learning of scheduling weights*: producing the probe path schedule based on inspection rules requires the proper weighting of rules when they conflict. A separate associative memory, such as the EAM, can

be used to learn the inspection weights in relationship to the part-model and tolerance callouts. When a similar part-model is encountered. the same weights, or a linear approximation of the weights, can be used when creating the rule matrices.

*Evaluation functions*: currently the evaluation functions are performed within the CMM language, CMES, which usually entails least squares algorithms. More efficient algorithms can operate upon the measurement points which will result in more accurate tolerance evaluation.

*Five dof probe*: extending the inspection planning to five dof will involve a large undertaking, but will produce better results. The point placement techniques, MR sequencer, and collision avoidance algorithm must all be modified.

*Representing fixtures in simulation and collision avoidance*: solid fixture structures will provide a safer path for the inspector. In addition, intelligent fixture selection and placement will aid the inspector and produce more options for setup orientation selection.

*More human interaction*: yes, more. Allowing human interaction into key areas of the IPEM will speed the process and guide it into a desired result. For example, the current setup selection is a computationally expensive algorithm: however, this can be replaced by an optional resting-surface selection by the inspector.

# References

Copy from one, it's plagiarism;
copy from two, it's research.
- Wilson Mizner

Abe, Shigeo., Junzo Kawakami, and Kotarro Hirasawa, 1992. "Solving Inequality Constrained Combinatorial Optimization Problems by the Hopfield Neural Networks," *Neural Networks, 5,* 663-670.

Adorf, H.-M., 1989. "Knowledge-Based Systems in Astronomy," A. Heck and F. Murtagh (eds.), Lecture Notes in Physics, *329,* 251-245.

American Society of Mechanical Engineers. (1982) *Dimensioning and Tolerancing, ANSI Y14.5M,* New York: American Society of Mechanical Engineers.

Asada, H., and J.-J. E. Slotine, 1986. *Robot Analysis and Control.* New York: John Wiley & Sons, Inc.

Barraquand, Jérôme, 1991. "Automatic Motion Planning for Complex Articulated Bodies." Lecture notes presented at the ACM SIGGRAPH'91 course C28: *Motion Synthesis, Planning, and Control.* Los Vegas, NV, July.

Booch, Grady, 1991. *Object Oriented Design with Applications.* Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.

Bonner, Susan., and Robert B. Kelley, 1990. "A Novel Representation for Planning 3-D Collision-Free Paths," *IEEE Transactions on System, Man., and Cybernetics, 20, 6,* 1337-1351.

Brady, M. et al. eds., 1982. *Robot Motion.* Cambridge, MA: MIT Press.

Brown, Curtis. 1983. *Functional Architecture of a Generative Inspection Process Planner for Coordinate Measuring Machines.* Master's Thesis. Oklahoma State University.

Brown, Curtis. 1990. "IPPEX: An Automated Planning System for Dimensional Inspection," *CIRP, The International Institution for Production Engineering Research,* June.

Chang, Tien-Chien, and Richard A. Wysk, 1985. *An Introduction to Automated Process Planning Systems.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Canny, John. F., 1988. *The Complexity of Robot Motion Planning.* Cambridge, MA: The MIT Press.

Dorney, Timothy D., 1992. *Learned Adaptive Environment Control Using Artificial Neural Networks and Fuzzy Logic.* Master's Thesis. Case Western Reserve University.

ElMaraghy, H. A., Gu, H. P., 1987. "Expert System for Inspection Planning," *Ann. of CIRP,* 85-89.

Etesami, Faryar, and Hong Qiao, 1989. "Analysis of Two-dimensional Measurement Data for Automated Inspection," *Journal of Manufacturing Systems,* 9, 1, 21-34.

Foster, Lowell W., 1986. "Geo-Metrics II - The Application of Geometric Tolerancing Techniques," Reading, MA: Addison-Wesley Publishing Co., Inc.

Galm, James M., 1991. *Functional Surface Metrology of Machined Parts.* Ph.D. Dissertation. Case Western Reserve University.

Gewali, Laxmi P., Simeon Ntafos, and Ioannis G. Tollis, 1990. "Path Planning in the Presence of Vertical Obstacles." *IEEE Transactions on Robotics and Automation.* 6, 3, 331-341.

Hayes, Caroline C., 1991. *Machining Planning: A Model of an Expert Level Planning Process.* Ph.D. Dissertation. Carnegie Mellon University.

Hopfield, J. J., and D. W. Tank, 1985. "'Neural' Computation of Decisions in Optimization Problems" *Biological Cybernetics,* 52, 141-152.

Hopp, Theodore H., and K. C. Lau, 1985. "A Hierarchical Model-Based Control System for Inspection" *Automated Manufacturing,* ASTM STP 862, 169-187.

Jeon, Okjune, 1990. *Efficient Inspection Path Planning.* Master's Thesis. Case Western Reserve University.

Johnston, M. D., and H.-M. Adorf, 1992. "Scheduling with Neural Networks — The Case of the Hubble Space Telescope." *Computers Operations Research,* 19, 3/4, 209-240.

Johnston, M. D., and H.-M. Adorf. 1989. "Learning in Stochastic Neural Networks for Constraint Satisfaction Problems." *Proceedings NASA Conference on Space Telerobotics.* 31 Jan. - 2 Feb., Pasadena, CA. G. Rodriguez and H. Seraji (eds.), JPL Publ. 89-7, *II,* 367-376.

Joshi, Sanjay, and Tien-Chien Chang, 1990. "Feature Extraction and Feature Based Design Approaches in the Development of Design Interfaces for Process Planning," *Journal of Intelligent Manufacturing,* 1, 1-15.

Keene, Sonya E.. 1989. *Object Oriented Programming in Common LISP*. Reading, MA: Addison Wesley.

Kosiba, Eric D., Jeff R. Wright, and Arthur E. Cobbs, 1992. "Discrete Event Sequencing as a Traveling Salesman Problem," *Computers in Industry*, *19*, 317-327.

Kosko, Bart, 1992. *Neural Networks and Fuzzy Systems - A Dynamic System Approach to Machine Intelligence*. New York: Prentice-Hall, Inc.

Laporte, Gilbert, 1992. "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms," *European Journal of Operational Research*, *59*, 231-247.

Latombe, Jean-Claude, 1991. *Robot Motion Planning*. Boston, MA: Kluwer Academic Press.

LeClair, Steven R., 1991. "The Rapid Design System: Memory-Driven Feature-Based Design," *Proceedings of the 1991 IEEE Conference on System Engineering*, Aug. 3, Dayton. OH. 35-37.

Liou, F. W.. and D. J. Suen, 1991. "The Development of a Feature-Based Fixture Process Planning System for Flexible Assembly" *Journal of Manufacturing Systems*, *11*, 2, 102-112.

LK Tool, Inc.. 1985. *LK CMES Inspection Language Reference Manual*.

Logee, Stephen, 1989. "Streamlining CMM Part Programming," *Quality*, Sep.

Lozano-Perez, T., 1987. "A Simple Motion-Planning Algorithm for General Robot Manipulators," *IEEE Journal of Robotics and Automation*, *RA-3*, 3, 224-238.

Lozano-Perez, T.. and M. A. Wesley, 1979. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of the ACM*, *22*, 10. 165-175.

Madsen, David. A., 1988. *Geometric Dimensioning and Tolerancing*. South Holland, IL.: The Goodheart-Willcox Co., Inc.

Magill, W. R., and A. J. McLeod. 1988. "Automated Generation of NC Part Programs from a Feature-based Component Description," *International Journal of Computer Integrated Manufacturing*, *2*, 4, 194-204.

Menq, Chia-Hsiang, and Hong-Tzong Yau. 1991a. "Path Planning for Automated Dimensional Inspection Using Coordinate Measuring Machines" *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*. Sacramento, CA. 1934-1938.

Menq, Chia-Hsiang, and Hong-Tzong Yau. 1992a. "An Automated Dimensional Inspection Environment for Manufactured Parts Using Coordinate Measuring Machines." *International Journal of Production Research*, *30*, 7. 1517-1536.

Menq, Chia-Hsiang., Hong-Tzong Yau, and Ching-Li Wing, 1991b. "An Intelligent Planning Environment for Automated Dimensional Inspection Using Coordinate Measurement Machines," *Transactions of the ASME, 114*, 222-230.

Menq, Chia-Hsiang., Hong-Tzong Yau. and Ching-Li Wing, 1992b. "Automated Precision Measurement of Surface Profiles in CAD-Directed Inspection," *IEEE Transactions on Robotics and Automation, 8*, 268-278.

Merat, Francis L., and Gerald M. Radack, 1992. "Automatic Inspection Planning Within a Feature-Based CAD System," *Robotics & Computer-Integrated Manufacturing, 9*, 1, 61-69.

Merat, F. L., G. M. Radack, K. Roumina, and S. M. Ruegsegger, 1991. "Automated Inspection Planning Within the Rapid Design System," *Proceedings of the 1991 IEEE International Conference on Systems Engineering*, Dayton, OH. 42-48.

Pao, Yoh-Han, 1989. *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley.

Pao, Yoh-Han, Kambiz Komeyli, Tanvir Goraya, and Steven LeClair, 1991. "A Computer-Based Adaptive Associative Memory in Support of Design and Planning," *Proceedings of the 1991 IEEE Conference on System Engineering*, Aug. 3, Dayton, OH. 49-55.

Pratt, I., 1991. "An Algorithm for Planning `Sensible´ Routes," *Engineering Application of Artificial Intelligence, 4*, 2, 97-108.

Prepatata, F. P. and S. J. Hong, "Convex Hull of Finite Sets of Points in Two and Three Dimensions." *Communications of the ACM, 20*, 2, Feb. 87-93.

Radack, Gerald M., and Francis L. Merat. 1990. "The Integration of Inspection into the CIM Environment" *Proceedings of the Hawaiian International Conference on System Science*, Jan.

Radack, Gerald M.. Jeremy F. Jacobsohn. and Francis L. Merat. 1991. "The Rapid Design System: Memory-Driven Feature-Based Design," *Proceedings of the 1991 IEEE Conference on System Engineering*, Aug. 3, Dayton, OH. 38-41.

Schreiber, Rita R.. 1990. "The Methods Divergence Dilemma." *Manufacturing Engineering*, May, 10.

Sharir, Micha.. and Amir Schorr, 1986. "On Shortest Paths in Polyhedral Space," *American Journal of Computing, 15*, 1, 193-215.

Sprow, Eugene, 1990. "Challenges to CMM Precision," *Tooling and Production*. Nov, 54-61.

Spyridi, A. J., and A. A. G. Requicha, 1990. "Accessibility Analysis for the Automatic Inspection of Mechanical Parts by Coordinate Measuring Machines." *IEEE*, 1284-1289.

Supinski, Mark R., Pius J. Egbelu, and El-Amine Lehtihet. 1991. "Automatic Plan and Robot Code Generation for PCB Assembly," *Manufacturing Review*, *4*, 3, 214-224.

Takefuji, Yoshiyasu, 1992. *Neural Network Parallel Computing*. Boston, MA: Kluwer Academic Publishers.

Touretzky, David S., and Geoffrey E. Hinton, 1988. "A Distributed Connectionist Production System," *Cognitive Science*, *12*, 423-466.

Traband, Mark T., and D. J. Medeiros, 1988. "CAD-directed Programming of a Vision-Based Inspection System," *Journal of Manufacturing Systems*, *8*, 3, 215-223.

Westhoven, Timothy. E., 1991. *Feature Sequencing in Process Plannnig using an Episodal Associative Memory*. Master's Thesis. Wright State University.

Xu, X., and W. T. Tsai, 1991. "Effective Neural Algorithms for the Traveling Salesman Problem," *Neural Networks*, *4*, 193-205.

Young, Robert E., Authur Greef, and Peter O'Grady, 1992. "An Artificial intelligence-based Constraint Network System for Concurrent Engineering," *International Journal of Product Research*. *30*, 7, 1715-1735.

# Appendix A

## CMES Macros for Design/Tolerance Feature Combinations

---

### BOSS

| tol | CMES macro |
|-----|-----------|

⊕
```
#OD,axis///x\y\z\diam,0,0\tol
#MC,axis\clear_val
```

⊥
```
#OD,axis\x\y\zi\diam          \SP,1
#MC,axis\clear_val
#OD,axis\x\y\z2\diam\SP,2
#MC,axis\clear_val
UP,1,2\DI,4
{ AQ,4,L,axis./DL\tol }
{ AQ,4,l,axis,P\tol }
```

//
```
SA,10
#OD,axis1\x1\y1\z1a\diam1  \SP,1
#OD,axis1\x1\y1\z1b\diam1  \SP,2
#MC,axis1\clear_val
UP,1,2\DI,axis
! ** note origin has been shifted to axis of boss
#OD,axis2\x2\y2\z2a\diam2  \SP,1
#OD,axis2\x2\y2\z2b\diam2  \SP,2
#MC,axis2\clear_val
UP,1,2\DI,4
{ AP,4,L,axis./DL\tol }
{ AP,4,L,axis,P\tol }
RA,10
```

—  #OD,axis\x1\y1\z\diam        \SP,1

#MC,axis\clear_val
{ ... n total times (1 line): 2 _ n _ 40 ... }
UG,1,n\DI,4,n
{ FS,4\tol }
{ FS,4,axis,P\tol }


o  SA,10

#OD,axis\x\y\z1\diam1
#MC,axis\clear_val
DI,4
#OD,axis\x\y\z2\diam2
#MC,axis\clear_val
CN,4,L,axis,L,/D\tol
RA,10


∠  ! for angles other than 90 deg

#MC,axis\x\y\z
#PP,axis\x\y\z              \SP,1
{ repeat at least n > 2 times }
UG,1,n\DI.4
{ AA,4,L,datum,/D }
{ AA,4,L,datum,/L }


axis-D  #OD,axis\x\y\z1\diam        \SP,1
#MC,axis\clear_val
#OD,axis.x,y,z2\diam        \SP,2
#MC,axis\clear_val
UP,1,2\DI,axis_name



## BLIND/THROUGH HOLE
tol    CMES macro

❖  #ID,axis///\x\y\z\diam,0,0\tol

#MC,axis\clear_val

⊥  #ID,axis\x\y\z1\diam  \SP,1

#MC,axis\clear_val
#ID,axis\x\y\z2\diam  \SP,2
#MC,axis\clear_val
UP,1,2\DI,4
{ AQ,4,L,axis,/DL\tol }
{ AQ,4,l,axis,P\tol }


//  SA,10

#ID,axis1\x1\y1\z1a\diam1 \SP,1
#ID,axis1\x1\y1\z1b\diam1 \SP,2
#MC,axis1\clear_val
UP,1,2\DI,axis
! ** note that origin has been shifted to the axis of the hole
#ID,axis2\x2\y2\z2a\diam2 \SP,1
#ID,axis2\x2\y2\z2b\diam2 \SP,2
#MC,axis2\clear_val
UP,1,2\DI,4
{ AP,4,L,axis,/DL\tol }
{ AP,4,L,axis,P\tol }
RA,10


—  #ID,axis\x1\y1\z\diam  \SP,1

{ ... n total times (1 line): 2 _ n _ 40 ... }
UG,1..n\DI,4,n/
{ FS,4,axis,P\tol }
{ FS,4\tol }


●  SA,10

#ID,axis\x\y\z1\diam1
#MC,axis\clear_val
DI,4
#ID,axis\x\y\z2\diam2
#MC,axis\clear_val
CN,4,L,axis,L,/D\tol
RA,10


∠  ! for angles other than 90 deg

```
       #MC,axis\x\y\z
       #PP,axis\x\y\z               \SP,1
       { repeat at least n > 2 times }
       UG,1,n\DI,4
       { AA,4,L,datum./D }
       { AA,4,L,datum./L }


axis-D  #ID,axis\x\y\z1\diam        \SP,1
        #MC,axis\clear_val
        #ID,axis\x\y\z2\diam        \SP,2
        #MC,axis\clear_val
        UP,1,2\DI,axis_name
```

## EDGE CUT
tol     CMES macro

∠       ! assume datum is already created

```
        #MC\x1\y1\z1
        #PP,axis\z0              \SP,1
        { ... n total times (2 lines): 3 ≤ n ≤ 40 ... }
      · UG,1,n\AX,4,n
        AA,4,P,axis,n/Paxis\angle\tol
```

◈       
```
        #MC\x1\y1\z1
        #PP,axis\z0     \SP,1
        { ... n total times (2 lines): 3 ≤ n ≤ 40 ... }
        UG,1,n\AX,4,n
        N1,4,datum/\tol(.5)
        ! AN,4,datum/\90,tol(.5)
```

## THROUGH SLOT
tol     CMES macro

⊥       #MC\x1\y1\z1
```
        #PP,axis\y0             \SP,1
        { ... n total times (2 lines): 6 ≤ n ≤ 40 ... }
        UP,1,2\CM,axis          \SP,1
```

```
UP,3,4\CM,axis        \SP,2
UP,5,6\CM,axis        \SP,3
{ ... n total times (1 line): 3 ≤ n ≤ 20 ... }
UG,1,n\AX,4,n
N1,4,axis\tol(.5)
! AN,4,datum\90,tol(.5)
```

## PLANAR SURFACES

tol     CMES macro

◆     #MC\x1\y1\z1

```
#PP,axis\z0    \SP,1
{ ... n total times (2 lines): 3 _ n _ 40 ... }
UG,1,n\AX,4,n
N1,4,datum\tol(.5)
! or AN,4,datum\90,tol(.5)
```

//     #MC\x1\y1\z1

```
#PP,axis\z0    \SP,1
{ ... n total times (2 lines): 3 ≤ n ≤ 40 ... }
UG,1,n\AX,4,n
AP,4,P,axis,P,/P\tol
```

—     #MC\x1\y1\z1

```
#PP,axis\z0    \SP,1
{ ... n total times (2 lines): 2 ≤ n ≤ 40 ... }
UG,1,n\AX+,4
FS,4,P,axis,P\tol
{ repeat above code for each new line }
```

∠     #MC\x1\y1\z1

```
#PP,axis\z0    \SP,1
{ ... n total times (2 lines): 3 ≤ n ≤ 40 ... }
UG,1,n\AX,4
AA,4,P,axis,n/Paxis\angle\tol
```

⧄     #MC\x1\y1\z1

```
#PP,axis\p0    \SP,1
```

{ ... n total times (2 lines): $3 \leq n \leq 40$ ... }
FF,axis,/P\tol

# Appendix B

# CMES Quick Reference

| Command | Description |
| --- | --- |
| AX | Axis. The normal of the plane containing the points given is created. If more than three points are given, a least-squares algorithm is used to compute the plane. |
| N1 | Normal 1. This creates the normal of a plane from 2 measured points. The third point is calculated from the normal of a specified plane. This ensures mathematical mutual perpendicularity needed for datum reference frames. |
| N2 | Normal 2. This creates the normal of a plane from 1 measured point. The second and third points are calculated from the normal of two specified planes. |
| PI | Point of Intersection. This command determines the point of intersection of that have been created by a combination of AX, N1, and N2 commands. |
| MD | Master Datum. Follows the MD command to make the intersection point the workpiece origin. |
| #MC | Automatic Move Course. The # signifies that the command will physically move the probe, and collision avoidance techniques need to be applied. The MC command is a fast move and needs to be well clear of the part. |
| #PP | Automatic Point with Probe Compensation. This command moves the probe slowly in one direction to take a measurement from the part. |

#ID Automatic Inside Diameter. Inspects a hole by accelerating to the point above the hole, enter the hole, and probe the inside of the hole.

#OD Automatic Outside Diameter. Inspects a boss in the same manner as a hole.

/// Tolerances. Backslashes are appended to end of any evaluation command to indicate tolerancing. Equal bilateral: / = ±val, Unequal bilateral: // = hival, loval. True position: /// = tp-diameter, True Position plus bonus: M = tp-diameter.

SP Save Point. Save a point for future use.

UP Use Point. Use a saved point.

FS Form - Straightness. Determines the straightness of a line.

FF Form - Flatness. Determines the flatness of a surface.

AP Attitude - Parallelism. Determines the parallelism of one feature to one (or two) other features.

AQ Attitude - Squareness. Determines the perpendicularity of one feature (or two) to another.

AA Attitude - Angularity. Determines the angularity of one feature to another.

CN Concentricity. Determines the concentricity of a point to a line, line to a point, or line to a line.

# Appendix C

# Artificial Neural Net Code in C

```c
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <fcntl.h>
#define DIM 20
#define UL 500.0
#define LL -500.0

void fpm_sol(int[][DIM],float[][DIM],char**);
int in_dim;
main(int argc, char *argv[])
{
 int i, j, k, z, seed, maxiter=100000, v[DIM][DIM], th, iter, sp, sf2;
 int a1=1, a2=1, b=2, c=2, d1=1, d2=1, s1=0, s2=-1, S[DIM][DIM];
 int time_step=1, p, p1, p2, sa1, sa2[DIM], h1, h2[DIM];
 float f1=1.0, sf, sb, sc, dis, min, max;
 float du[DIM][DIM], u[DIM][DIM], d[DIM][DIM], dd[DIM][DIM], F[DIM][DIM];

 Read_weights(argv[1], d, F, S);
 sp=0;
/* minimum shift wieghts */
 for(i=0;i<in_dim;i++) {
  min=1000;
  for(j=0;j<in_dim;j++)
   if((i!=j)&&(d[i][j]<min)) min=d[i][j];
  for(j=0;j<in_dim;j++)
   if(i!=j) dd[i][j]=d[i][j]-min;
 }

/* initialize */
 iter = 0;
 for(i=0;i<in_dim;i++)
  for(j=0;j<in_dim;j++) {
   u[i][j] = -rand()/100000.0;
   v[i][j]=0;
  }
```

```
v[sp][0]=1; /* v[ep][in_dim-1]=1; */

while(iter<maxiter) {
 /* evaluate */
 for(i=0;i<in_dim;i++)
  if (i!=sp) /* &&(i!=ep) */
       for(j=1;j<in_dim;j++)
         if(u[i][j]>0)  v[i][j]=1;
         else      v[i][j]=0;

 /* rules manipulation */
 if (s1 != 0)
  for (i=1; i<in_dim: i++)
       for (j=1; j<in_dim: j++)
        v[i][j] = v[i][j] && S[i][j];

 th = 0;
 sa2[0]=0;
 /* column summation */
 for(j=1;j<in_dim;j++) {
  sa2[j] = 0;  h2[j] = 0;
  for(p=0;p<in_dim;p++)
       sa2[j] = sa2[j]+v[p][j];
  if(sa2[j]==0) h2[j] = 1;
       sa2[j] = sa2[j]-1;
 }
 /* row summation */
 for(i=0;i<in_dim;i++)
 if (i!=sp)   {
  sa1 = 0;  h1 = 0;
  for(j=1;j<in_dim;j++)
       sa1 = sa1+v[i][j];
  if(sa1==0) h1 = 1;
  sa1 = sa1-1;

  for(j=1;j<in_dim;j++) {
       sb = 0; sf = 0;
       for(p=0;p<in_dim;p++) {
        if (p!=i) sb = sb+(v[p][j-1] + v[p][j+1])*dd[i][p];
        if (p!=i) sf = sf+(v[p][j-1]*F[p][i] + v[p][j+1]*F[i][p]);
        }

       sf2 = 0;
       for(k=1;k<in_dim;k++)
        if ((k!=i) && (F[i][k] < 0.0)) sf2 += v[i][j]*v[k][j+1];
       if (sf2 == 1) sf = 0.0;
       sf2 = 0;
       for(k=1;k<in_dim;k++)
        if ((k!=i) && (F[k][i] < 0.0)) sf2 += v[k][j-1]*v[i][j];
       if (sf2 == 1) sf = 0.0;
```

```c
                sc = 0;
                if((v[i][j]==1)&&(sa2[j-1]==0)&&(sa2[j+1]==0)) {
                  for(p=0;p<in_dim;p++)  {
                    if(v[p][j-1]==1)  p1=p;
                    if(v[p][j+1]==1)  p2=p;
                  }
                  if((p1!=i)&&(p2!=i)&&(p1!=p2))
                    sc = dd[i][p1]+dd[i][p2]-dd[p2][p1]-dd[p1][p2];
                }
                th = th + abs(sa1) + abs(sa2[j]);
                /* neuron motion equation */
                du[i][j] = -a1*sa1-a2*sa2[j] - b*sb - c*sc + d1*h1+d2*h2[j] - f1*sf - s2*S[i][j];
                /* update */
                u[i][j] = u[i][j]+du[i][j]*time_step;
                /* threshold limits */
                if (u[i][j] > UL) u[i][j] = UL;
                if (u[i][j] < LL) u[i][j] = LL;
          }
        }

    if(th==0) break;
    iter=iter+1;
  }
  fprn_sol(v, d, argv);
}

void fprn_sol(int v[][DIM], float d[][DIM], char *outfile[])
{
 int i,j;
 char fname[24];
 FILE *fp;
 strcpy(fname, strcat(outfile[1],".out"));
 printf("\n\nPrinting to \"%s\" ", fname);
 fp = fopen(fname, "w");
 for(j=0;j<in_dim;j++)
   for(i=0;i<in_dim;i++)
     if (v[i][j] == 1) {
          fprintf(fp,"%.0f %.0f ",d[in_dim][i], d[in_dim+1][i]);
          printf("%.0f %.0f, ",d[in_dim][i], d[in_dim+1][i]);
     }
 fclose(fp);
}
```